

JURASSIC

Generated by Doxygen 1.9.1

<b>1 Main Page</b>	<b>1</b>
<b>2 Data Structure Index</b>	<b>1</b>
2.1 Data Structures	1
<b>3 File Index</b>	<b>2</b>
3.1 File List	2
<b>4 Data Structure Documentation</b>	<b>3</b>
4.1 atm_t Struct Reference	3
4.1.1 Detailed Description	4
4.1.2 Field Documentation	4
4.2 ctl_t Struct Reference	7
4.2.1 Detailed Description	9
4.2.2 Field Documentation	9
4.3 los_t Struct Reference	16
4.3.1 Detailed Description	17
4.3.2 Field Documentation	17
4.4 obs_t Struct Reference	20
4.4.1 Detailed Description	20
4.4.2 Field Documentation	21
4.5 ret_t Struct Reference	23
4.5.1 Detailed Description	24
4.5.2 Field Documentation	24
4.6 tbl_t Struct Reference	29
4.6.1 Detailed Description	29
4.6.2 Field Documentation	29
<b>5 File Documentation</b>	<b>31</b>
5.1 brightness.c File Reference	31
5.1.1 Detailed Description	31
5.1.2 Function Documentation	31
5.2 brightness.c	32
5.3 climatology.c File Reference	33
5.3.1 Detailed Description	33
5.3.2 Function Documentation	34
5.4 climatology.c	35
5.5 filter.c File Reference	36
5.5.1 Detailed Description	36
5.5.2 Function Documentation	36
5.6 filter.c	39
5.7 formod.c File Reference	40
5.7.1 Detailed Description	41
5.7.2 Function Documentation	41

5.8 formod.c . . . . .	46
5.9 hydrostatic.c File Reference . . . . .	50
5.9.1 Detailed Description . . . . .	50
5.9.2 Function Documentation . . . . .	50
5.10 hydrostatic.c . . . . .	51
5.11 interpolate.c File Reference . . . . .	52
5.11.1 Detailed Description . . . . .	52
5.11.2 Function Documentation . . . . .	52
5.12 interpolate.c . . . . .	53
5.13 invert.c File Reference . . . . .	54
5.13.1 Detailed Description . . . . .	55
5.13.2 Macro Definition Documentation . . . . .	55
5.13.3 Function Documentation . . . . .	55
5.14 invert.c . . . . .	59
5.15 jsec2time.c File Reference . . . . .	64
5.15.1 Detailed Description . . . . .	64
5.15.2 Function Documentation . . . . .	64
5.16 jsec2time.c . . . . .	65
5.17 jurassic.c File Reference . . . . .	65
5.17.1 Detailed Description . . . . .	68
5.17.2 Function Documentation . . . . .	68
5.18 jurassic.c . . . . .	150
5.19 jurassic.h File Reference . . . . .	215
5.19.1 Detailed Description . . . . .	220
5.19.2 Macro Definition Documentation . . . . .	221
5.19.3 Function Documentation . . . . .	232
5.20 jurassic.h . . . . .	315
5.21 kernel.c File Reference . . . . .	324
5.21.1 Detailed Description . . . . .	324
5.21.2 Function Documentation . . . . .	324
5.22 kernel.c . . . . .	326
5.23 limb.c File Reference . . . . .	327
5.23.1 Detailed Description . . . . .	327
5.23.2 Function Documentation . . . . .	327
5.24 limb.c . . . . .	328
5.25 nadir.c File Reference . . . . .	329
5.25.1 Detailed Description . . . . .	329
5.25.2 Function Documentation . . . . .	329
5.26 nadir.c . . . . .	330
5.27 obs2spec.c File Reference . . . . .	331
5.27.1 Detailed Description . . . . .	331
5.27.2 Function Documentation . . . . .	331

5.28 obs2spec.c . . . . .	333
5.29 planck.c File Reference . . . . .	334
5.29.1 Detailed Description . . . . .	334
5.29.2 Function Documentation . . . . .	334
5.30 planck.c . . . . .	335
5.31 raytrace.c File Reference . . . . .	336
5.31.1 Detailed Description . . . . .	336
5.31.2 Function Documentation . . . . .	337
5.32 raytrace.c . . . . .	338
5.33 retrieval.c File Reference . . . . .	340
5.33.1 Detailed Description . . . . .	341
5.33.2 Function Documentation . . . . .	341
5.34 retrieval.c . . . . .	354
5.35 tblfmt.c File Reference . . . . .	366
5.35.1 Detailed Description . . . . .	366
5.35.2 Function Documentation . . . . .	366
5.36 tblfmt.c . . . . .	367
5.37 tblgen.c File Reference . . . . .	368
5.37.1 Detailed Description . . . . .	368
5.37.2 Macro Definition Documentation . . . . .	368
5.37.3 Function Documentation . . . . .	368
5.38 tblgen.c . . . . .	370
5.39 time2jsec.c File Reference . . . . .	372
5.39.1 Detailed Description . . . . .	372
5.39.2 Function Documentation . . . . .	372
5.40 time2jsec.c . . . . .	373
<b>Index</b>	<b>375</b>

## 1 Main Page

The JUelich RApid Spectral Simulation Code (JURASSIC) is a fast radiative transfer model for the mid-infrared spectral region. This reference manual provides information on the algorithms and data structures used in the code.

Further information can be found at: <https://github.com/slcs-jsc/jurassic>

## 2 Data Structure Index

### 2.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">atm_t</a>	Atmospheric data	3
<a href="#">ctl_t</a>	Forward model control parameters	7
<a href="#">los_t</a>	Line-of-sight data	16
<a href="#">obs_t</a>	Observation geometry and radiance data	20
<a href="#">ret_t</a>	Retrieval control parameters	23
<a href="#">tbl_t</a>	Emissivity look-up tables	29

## 3 File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

<a href="#">brightness.c</a>	Convert radiance to brightness temperature	31
<a href="#">climatology.c</a>	Prepare atmospheric data file from climatological data	33
<a href="#">filter.c</a>	Create radiometric filter functions	36
<a href="#">formod.c</a>	JURASSIC forward model	40
<a href="#">hydrostatic.c</a>	Recalculate pressure based on hydrostatic equilibrium	50
<a href="#">interpolate.c</a>	Interpolate atmospheric data to another spatial grid	52
<a href="#">invert.c</a>	Inversion tool for MPTRAC	54
<a href="#">jsec2time.c</a>	Convert Julian seconds to date	64
<a href="#">jurassic.c</a>	JURASSIC library definitions	65
<a href="#">jurassic.h</a>	JURASSIC library declarations	215
<a href="#">kernel.c</a>	Calculate kernel functions	324

<a href="#">limb.c</a>	Create observation geometry for a limb sounder	327
<a href="#">nadir.c</a>	Create observation geometry for a nadir sounder	329
<a href="#">obs2spec.c</a>	Converter for spectra	331
<a href="#">planck.c</a>	Convert brightness temperature to radiance	334
<a href="#">raytrace.c</a>	Determine atmospheric ray paths	336
<a href="#">retrieval.c</a>	JURASSIC retrieval processor	340
<a href="#">tblfmt.c</a>	Convert look-up table file format	366
<a href="#">tblgen.c</a>	Prepape look-up tables from monochromatic absorption spectra	368
<a href="#">time2jsec.c</a>	Convert date to Julian seconds	372

## 4 Data Structure Documentation

### 4.1 atm\_t Struct Reference

Atmospheric data.

```
#include <jurassic.h>
```

#### Data Fields

- int [np](#)  
*Number of data points.*
- double [time](#) [NP]  
*Time (seconds since 2000-01-01T00:00Z).*
- double [z](#) [NP]  
*Altitude [km].*
- double [lon](#) [NP]  
*Longitude [deg].*
- double [lat](#) [NP]  
*Latitude [deg].*
- double [p](#) [NP]  
*Pressure [hPa].*
- double [t](#) [NP]  
*Temperature [K].*
- double [q](#) [NG][NP]  
*Volume mixing ratio [ppv].*

- double `k` [NW][NP]  
*Extinction [1/km].*
- double `clz`  
*Cloud layer height [km].*
- double `cldz`  
*Cloud layer depth [km].*
- double `clk` [NCL]  
*Cloud layer extinction [1/km].*
- double `sfz`  
*Surface height [km].*
- double `sfp`  
*Surface pressure [hPa].*
- double `sft`  
*Surface temperature [K].*
- double `sfe` [NSF]  
*Surface emissivity.*

#### 4.1.1 Detailed Description

Atmospheric data.

Definition at line 343 of file [jurassic.h](#).

#### 4.1.2 Field Documentation

##### 4.1.2.1 `np` `int atm_t::np`

Number of data points.

Definition at line 346 of file [jurassic.h](#).

##### 4.1.2.2 `time` `double atm_t::time[NP]`

Time (seconds since 2000-01-01T00:00Z).

Definition at line 349 of file [jurassic.h](#).

##### 4.1.2.3 `z` `double atm_t::z[NP]`

Altitude [km].

Definition at line 352 of file [jurassic.h](#).

**4.1.2.4 lon** double atm\_t::lon[[NP](#)]

Longitude [deg].

Definition at line [355](#) of file [jurassic.h](#).

**4.1.2.5 lat** double atm\_t::lat[[NP](#)]

Latitude [deg].

Definition at line [358](#) of file [jurassic.h](#).

**4.1.2.6 p** double atm\_t::p[[NP](#)]

Pressure [hPa].

Definition at line [361](#) of file [jurassic.h](#).

**4.1.2.7 t** double atm\_t::t[[NP](#)]

Temperature [K].

Definition at line [364](#) of file [jurassic.h](#).

**4.1.2.8 q** double atm\_t::q[[NG](#)] [[NP](#)]

Volume mixing ratio [ppv].

Definition at line [367](#) of file [jurassic.h](#).

**4.1.2.9 k** double atm\_t::k[[NW](#)] [[NP](#)]

Extinction [1/km].

Definition at line [370](#) of file [jurassic.h](#).



**4.1.2.10 clz** `double atm_t::clz`

Cloud layer height [km].

Definition at line 373 of file [jurassic.h](#).

**4.1.2.11 cldz** `double atm_t::cldz`

Cloud layer depth [km].

Definition at line 376 of file [jurassic.h](#).

**4.1.2.12 clk** `double atm_t::clk`[\[NCL\]](#)

Cloud layer extinction [1/km].

Definition at line 379 of file [jurassic.h](#).

**4.1.2.13 sfz** `double atm_t::sfz`

Surface height [km].

Definition at line 382 of file [jurassic.h](#).

**4.1.2.14 sfp** `double atm_t::sfp`

Surface pressure [hPa].

Definition at line 385 of file [jurassic.h](#).

**4.1.2.15 sft** `double atm_t::sft`

Surface temperature [K].

Definition at line 388 of file [jurassic.h](#).

**4.1.2.16 sfeps** double atm\_t::sfeps[NSF]

Surface emissivity.

Definition at line 391 of file [jurassic.h](#).

The documentation for this struct was generated from the following file:

- [jurassic.h](#)

**4.2 ctl\_t Struct Reference**

Forward model control parameters.

```
#include <jurassic.h>
```

**Data Fields**

- int [ng](#)  
Number of emitters.
- char [emitter](#) [NG][LEN]  
Name of each emitter.
- int [nd](#)  
Number of radiance channels.
- double [nu](#) [ND]  
Centroid wavenumber of each channel [ $\text{cm}^{-1}$ ].
- int [nw](#)  
Number of spectral windows.
- int [window](#) [ND]  
Window index of each channel.
- int [ncl](#)  
Number of cloud layer spectral grid points.
- double [clnu](#) [NCL]  
Cloud layer wavenumber [ $\text{cm}^{-1}$ ].
- int [nsf](#)  
Number of surface layer spectral grid points.
- double [sfnu](#) [NSF]  
Surface layer wavenumber [ $\text{cm}^{-1}$ ].
- int [sftype](#)  
Surface treatment (0=none, 1=emissions, 2=downward, 3=solar).
- double [sfsza](#)  
Solar zenith angle at the surface [deg] (-999=auto).
- char [tblbase](#) [LEN]  
Base name for table files and filter function files.
- int [tblfmt](#)  
Look-up table file format (1=ASCII, 2=binary).
- double [hydz](#)  
Reference height for hydrostatic pressure profile (-999 to skip) [km].
- int [ctm\\_co2](#)  
Compute CO2 continuum (0=no, 1=yes).

- int `ctm_h2o`  
*Compute H2O continuum (0=no, 1=yes).*
- int `ctm_n2`  
*Compute N2 continuum (0=no, 1=yes).*
- int `ctm_o2`  
*Compute O2 continuum (0=no, 1=yes).*
- int `refrac`  
*Take into account refractivity (0=no, 1=yes).*
- double `rayds`  
*Maximum step length for raytracing [km].*
- double `raydz`  
*Vertical step length for raytracing [km].*
- char `fov` [`LEN`]  
*Field-of-view data file.*
- double `retp_zmin`  
*Minimum altitude for pressure retrieval [km].*
- double `retp_zmax`  
*Maximum altitude for pressure retrieval [km].*
- double `rett_zmin`  
*Minimum altitude for temperature retrieval [km].*
- double `rett_zmax`  
*Maximum altitude for temperature retrieval [km].*
- double `retq_zmin` [`NG`]  
*Minimum altitude for volume mixing ratio retrieval [km].*
- double `retq_zmax` [`NG`]  
*Maximum altitude for volume mixing ratio retrieval [km].*
- double `retk_zmin` [`NW`]  
*Minimum altitude for extinction retrieval [km].*
- double `retk_zmax` [`NW`]  
*Maximum altitude for extinction retrieval [km].*
- int `ret_clz`  
*Retrieve cloud layer height (0=no, 1=yes).*
- int `ret_cldz`  
*Retrieve cloud layer depth (0=no, 1=yes).*
- int `ret_clk`  
*Retrieve cloud layer extinction (0=no, 1=yes).*
- int `ret_sfz`  
*Retrieve surface layer height (0=no, 1=yes).*
- int `ret_sfp`  
*Retrieve surface layer pressure (0=no, 1=yes).*
- int `ret_sft`  
*Retrieve surface layer temperature (0=no, 1=yes).*
- int `ret_sfeps`  
*Retrieve surface layer emissivity (0=no, 1=yes).*
- int `write_bbt`  
*Use brightness temperature instead of radiance (0=no, 1=yes).*
- int `write_matrix`  
*Write matrix file (0=no, 1=yes).*
- int `formod`  
*Forward model (1=EGA, 2=RFM).*
- char `rfmbin` [`LEN`]

- Path to RFM binary.*
  - char `rfmhit` [`LEN`]  
*HITRAN file for RFM.*
- char `rfmxsc` [`NG`][`LEN`]  
*Emitter cross-section files for RFM.*

### 4.2.1 Detailed Description

Forward model control parameters.

Definition at line 396 of file [jurassic.h](#).

### 4.2.2 Field Documentation

#### 4.2.2.1 `ng` `int` `ctl_t::ng`

Number of emitters.

Definition at line 399 of file [jurassic.h](#).

#### 4.2.2.2 `emitter` `char` `ctl_t::emitter`[`NG`][`LEN`]

Name of each emitter.

Definition at line 402 of file [jurassic.h](#).

#### 4.2.2.3 `nd` `int` `ctl_t::nd`

Number of radiance channels.

Definition at line 405 of file [jurassic.h](#).

#### 4.2.2.4 `nu` `double` `ctl_t::nu`[`ND`]

Centroid wavenumber of each channel [ $\text{cm}^{-1}$ ].

Definition at line 408 of file [jurassic.h](#).

**4.2.2.5** `nw` `int` `ctl_t::nw`

Number of spectral windows.

Definition at line [411](#) of file [jurassic.h](#).

**4.2.2.6** `window` `int` `ctl_t::window`[[ND](#)]

Window index of each channel.

Definition at line [414](#) of file [jurassic.h](#).

**4.2.2.7** `ncl` `int` `ctl_t::ncl`

Number of cloud layer spectral grid points.

Definition at line [417](#) of file [jurassic.h](#).

**4.2.2.8** `clnu` `double` `ctl_t::clnu`[[NCL](#)]

Cloud layer wavenumber [ $\text{cm}^{-1}$ ].

Definition at line [420](#) of file [jurassic.h](#).

**4.2.2.9** `nsf` `int` `ctl_t::nsf`

Number of surface layer spectral grid points.

Definition at line [423](#) of file [jurassic.h](#).

**4.2.2.10** `sfnu` `double` `ctl_t::sfnu`[[NSF](#)]

Surface layer wavenumber [ $\text{cm}^{-1}$ ].

Definition at line [426](#) of file [jurassic.h](#).

**4.2.2.11** `sftype` `int` `ctl_t::sftype`

Surface treatment (0=none, 1=emissions, 2=downward, 3=solar).

Definition at line 429 of file [jurassic.h](#).

**4.2.2.12** `sfsza` `double` `ctl_t::sfsza`

Solar zenith angle at the surface [deg] (-999=auto).

Definition at line 432 of file [jurassic.h](#).

**4.2.2.13** `tblbase` `char` `ctl_t::tblbase`[[LEN](#)]

Basename for table files and filter function files.

Definition at line 435 of file [jurassic.h](#).

**4.2.2.14** `tblfmt` `int` `ctl_t::tblfmt`

Look-up table file format (1=ASCII, 2=binary).

Definition at line 438 of file [jurassic.h](#).

**4.2.2.15** `hydzt` `double` `ctl_t::hydzt`

Reference height for hydrostatic pressure profile (-999 to skip) [km].

Definition at line 441 of file [jurassic.h](#).

**4.2.2.16** `ctm_co2` `int` `ctl_t::ctm_co2`

Compute CO2 continuum (0=no, 1=yes).

Definition at line 444 of file [jurassic.h](#).

**4.2.2.17 ctm\_h2o** `int ctl_t::ctm_h2o`

Compute H2O continuum (0=no, 1=yes).

Definition at line [447](#) of file [jurassic.h](#).

**4.2.2.18 ctm\_n2** `int ctl_t::ctm_n2`

Compute N2 continuum (0=no, 1=yes).

Definition at line [450](#) of file [jurassic.h](#).

**4.2.2.19 ctm\_o2** `int ctl_t::ctm_o2`

Compute O2 continuum (0=no, 1=yes).

Definition at line [453](#) of file [jurassic.h](#).

**4.2.2.20 refrac** `int ctl_t::refrac`

Take into account refractivity (0=no, 1=yes).

Definition at line [456](#) of file [jurassic.h](#).

**4.2.2.21 rayds** `double ctl_t::rayds`

Maximum step length for raytracing [km].

Definition at line [459](#) of file [jurassic.h](#).

**4.2.2.22 raydz** `double ctl_t::raydz`

Vertical step length for raytracing [km].

Definition at line [462](#) of file [jurassic.h](#).

**4.2.2.23** `fov` `char ctl_t::fov[LEN]`

Field-of-view data file.

Definition at line 465 of file [jurassic.h](#).

**4.2.2.24** `retp_zmin` `double ctl_t::retp_zmin`

Minimum altitude for pressure retrieval [km].

Definition at line 468 of file [jurassic.h](#).

**4.2.2.25** `retp_zmax` `double ctl_t::retp_zmax`

Maximum altitude for pressure retrieval [km].

Definition at line 471 of file [jurassic.h](#).

**4.2.2.26** `rett_zmin` `double ctl_t::rett_zmin`

Minimum altitude for temperature retrieval [km].

Definition at line 474 of file [jurassic.h](#).

**4.2.2.27** `rett_zmax` `double ctl_t::rett_zmax`

Maximum altitude for temperature retrieval [km].

Definition at line 477 of file [jurassic.h](#).

**4.2.2.28** `retq_zmin` `double ctl_t::retq_zmin[NG]`

Minimum altitude for volume mixing ratio retrieval [km].

Definition at line 480 of file [jurassic.h](#).



**4.2.2.29 retq\_zmax** `double ctl_t::retq_zmax` [\[NG\]](#)

Maximum altitude for volume mixing ratio retrieval [km].

Definition at line [483](#) of file [jurassic.h](#).

**4.2.2.30 retk\_zmin** `double ctl_t::retk_zmin` [\[NW\]](#)

Minimum altitude for extinction retrieval [km].

Definition at line [486](#) of file [jurassic.h](#).

**4.2.2.31 retk\_zmax** `double ctl_t::retk_zmax` [\[NW\]](#)

Maximum altitude for extinction retrieval [km].

Definition at line [489](#) of file [jurassic.h](#).

**4.2.2.32 ret\_clz** `int ctl_t::ret_clz`

Retrieve cloud layer height (0=no, 1=yes).

Definition at line [492](#) of file [jurassic.h](#).

**4.2.2.33 ret\_cldz** `int ctl_t::ret_cldz`

Retrieve cloud layer depth (0=no, 1=yes).

Definition at line [495](#) of file [jurassic.h](#).

**4.2.2.34 ret\_clk** `int ctl_t::ret_clk`

Retrieve cloud layer extinction (0=no, 1=yes).

Definition at line [498](#) of file [jurassic.h](#).

**4.2.2.35** `ret_sfz` `int ctl_t::ret_sfz`

Retrieve surface layer height (0=no, 1=yes).

Definition at line 501 of file [jurassic.h](#).

**4.2.2.36** `ret_sfp` `int ctl_t::ret_sfp`

Retrieve surface layer pressure (0=no, 1=yes).

Definition at line 504 of file [jurassic.h](#).

**4.2.2.37** `ret_sft` `int ctl_t::ret_sft`

Retrieve surface layer temperature (0=no, 1=yes).

Definition at line 507 of file [jurassic.h](#).

**4.2.2.38** `ret_sfeps` `int ctl_t::ret_sfeps`

Retrieve surface layer emissivity (0=no, 1=yes).

Definition at line 510 of file [jurassic.h](#).

**4.2.2.39** `write_bbt` `int ctl_t::write_bbt`

Use brightness temperature instead of radiance (0=no, 1=yes).

Definition at line 513 of file [jurassic.h](#).

**4.2.2.40** `write_matrix` `int ctl_t::write_matrix`

Write matrix file (0=no, 1=yes).

Definition at line 516 of file [jurassic.h](#).

**4.2.2.41 formod** `int ctl_t::formod`

Forward model (1=EGA, 2=RFM).

Definition at line 519 of file [jurassic.h](#).

**4.2.2.42 rfmbin** `char ctl_t::rfmbin[LEN]`

Path to RFM binary.

Definition at line 522 of file [jurassic.h](#).

**4.2.2.43 rfmbhit** `char ctl_t::rfmbhit[LEN]`

HITRAN file for RFM.

Definition at line 525 of file [jurassic.h](#).

**4.2.2.44 rfmxcsc** `char ctl_t::rfmxcsc[NG][LEN]`

Emitter cross-section files for RFM.

Definition at line 528 of file [jurassic.h](#).

The documentation for this struct was generated from the following file:

- [jurassic.h](#)

## 4.3 los\_t Struct Reference

Line-of-sight data.

```
#include <jurassic.h>
```

## Data Fields

- int **np**  
*Number of LOS points.*
- double **z** [NLOS]  
*Altitude [km].*
- double **lon** [NLOS]  
*Longitude [deg].*
- double **lat** [NLOS]  
*Latitude [deg].*
- double **p** [NLOS]  
*Pressure [hPa].*
- double **t** [NLOS]  
*Temperature [K].*
- double **q** [NLOS][NG]  
*Volume mixing ratio [ppv].*
- double **k** [NLOS][ND]  
*Extinction [1/km].*
- double **sft**  
*Surface temperature [K].*
- double **sfeps** [ND]  
*Surface emissivity.*
- double **ds** [NLOS]  
*Segment length [km].*
- double **u** [NLOS][NG]  
*Column density [molecules/cm<sup>2</sup>].*
- double **eps** [NLOS][ND]  
*Segment emissivity.*
- double **src** [NLOS][ND]  
*Segment source function [W/(m<sup>2</sup> sr cm<sup>-1</sup>)].*

### 4.3.1 Detailed Description

Line-of-sight data.

Definition at line 533 of file [jurassic.h](#).

### 4.3.2 Field Documentation

#### 4.3.2.1 np int los\_t::np

Number of LOS points.

Definition at line 536 of file [jurassic.h](#).

**4.3.2.2** **z** `double los_t::z[NLOS]`

Altitude [km].

Definition at line 539 of file [jurassic.h](#).

**4.3.2.3** **lon** `double los_t::lon[NLOS]`

Longitude [deg].

Definition at line 542 of file [jurassic.h](#).

**4.3.2.4** **lat** `double los_t::lat[NLOS]`

Latitude [deg].

Definition at line 545 of file [jurassic.h](#).

**4.3.2.5** **p** `double los_t::p[NLOS]`

Pressure [hPa].

Definition at line 548 of file [jurassic.h](#).

**4.3.2.6** **t** `double los_t::t[NLOS]`

Temperature [K].

Definition at line 551 of file [jurassic.h](#).

**4.3.2.7** **q** `double los_t::q[NLOS][NG]`

Volume mixing ratio [ppv].

Definition at line 554 of file [jurassic.h](#).

**4.3.2.8 k** `double los_t::k[NLOS]` [ND]

Extinction [1/km].

Definition at line 557 of file [jurassic.h](#).

**4.3.2.9 sft** `double los_t::sft`

Surface temperature [K].

Definition at line 560 of file [jurassic.h](#).

**4.3.2.10 sfeps** `double los_t::sfeps[ND]`

Surface emissivity.

Definition at line 563 of file [jurassic.h](#).

**4.3.2.11 ds** `double los_t::ds[NLOS]`

Segment length [km].

Definition at line 566 of file [jurassic.h](#).

**4.3.2.12 u** `double los_t::u[NLOS]` [NG]

Column density [molecules/cm<sup>2</sup>].

Definition at line 569 of file [jurassic.h](#).

**4.3.2.13 eps** `double los_t::eps[NLOS]` [ND]

Segment emissivity.

Definition at line 572 of file [jurassic.h](#).

#### 4.3.2.14 **src** double los\_t::src[NLOS] [ND]

Segment source function [ $W/(m^2 \text{ sr cm}^{-1})$ ].

Definition at line 575 of file [jurassic.h](#).

The documentation for this struct was generated from the following file:

- [jurassic.h](#)

## 4.4 **obs\_t** Struct Reference

Observation geometry and radiance data.

```
#include <jurassic.h>
```

### Data Fields

- int [nr](#)  
*Number of ray paths.*
- double [time](#) [NR]  
*Time (seconds since 2000-01-01T00:00Z).*
- double [obsz](#) [NR]  
*Observer altitude [km].*
- double [obslon](#) [NR]  
*Observer longitude [deg].*
- double [obslat](#) [NR]  
*Observer latitude [deg].*
- double [vpz](#) [NR]  
*View point altitude [km].*
- double [vplon](#) [NR]  
*View point longitude [deg].*
- double [vplat](#) [NR]  
*View point latitude [deg].*
- double [tpz](#) [NR]  
*Tangent point altitude [km].*
- double [tplon](#) [NR]  
*Tangent point longitude [deg].*
- double [tplat](#) [NR]  
*Tangent point latitude [deg].*
- double [tau](#) [ND][NR]  
*Transmittance of ray path.*
- double [rad](#) [ND][NR]  
*Radiance [ $W/(m^2 \text{ sr cm}^{-1})$ ].*

#### 4.4.1 Detailed Description

Observation geometry and radiance data.

Definition at line 580 of file [jurassic.h](#).

#### 4.4.2 Field Documentation

##### 4.4.2.1 `nr` `int obs_t::nr`

Number of ray paths.

Definition at line 583 of file [jurassic.h](#).

##### 4.4.2.2 `time` `double obs_t::time[NR]`

Time (seconds since 2000-01-01T00:00Z).

Definition at line 586 of file [jurassic.h](#).

##### 4.4.2.3 `obsz` `double obs_t::obsz[NR]`

Observer altitude [km].

Definition at line 589 of file [jurassic.h](#).

##### 4.4.2.4 `obslon` `double obs_t::obslon[NR]`

Observer longitude [deg].

Definition at line 592 of file [jurassic.h](#).

##### 4.4.2.5 `obslat` `double obs_t::obslat[NR]`

Observer latitude [deg].

Definition at line 595 of file [jurassic.h](#).

##### 4.4.2.6 `vpz` `double obs_t::vpz[NR]`

View point altitude [km].

Definition at line 598 of file [jurassic.h](#).



**4.4.2.7 vplon** `double obs_t::vplon[NR]`

View point longitude [deg].

Definition at line [601](#) of file [jurassic.h](#).

**4.4.2.8 vplat** `double obs_t::vplat[NR]`

View point latitude [deg].

Definition at line [604](#) of file [jurassic.h](#).

**4.4.2.9 tpz** `double obs_t::tpz[NR]`

Tangent point altitude [km].

Definition at line [607](#) of file [jurassic.h](#).

**4.4.2.10 tplon** `double obs_t::tplon[NR]`

Tangent point longitude [deg].

Definition at line [610](#) of file [jurassic.h](#).

**4.4.2.11 tplat** `double obs_t::tplat[NR]`

Tangent point latitude [deg].

Definition at line [613](#) of file [jurassic.h](#).

**4.4.2.12 tau** `double obs_t::tau[ND][NR]`

Transmittance of ray path.

Definition at line [616](#) of file [jurassic.h](#).

**4.4.2.13 rad** double obs\_t::rad[ND][NR]

Radiance [ $W/(m^2 \text{ sr cm}^{-1})$ ].

Definition at line 619 of file [jurassic.h](#).

The documentation for this struct was generated from the following file:

- [jurassic.h](#)

**4.5 ret\_t Struct Reference**

Retrieval control parameters.

**Data Fields**

- char [dir](#) [LEN]  
*Working directory.*
- int [kernel\\_recomp](#)  
*Re-computation of kernel matrix (number of iterations).*
- int [conv\\_itmax](#)  
*Maximum number of iterations.*
- double [conv\\_dmin](#)  
*Minimum normalized step size in state space.*
- int [err\\_ana](#)  
*Carry out error analysis (0=no, 1=yes).*
- double [err\\_formod](#) [ND]  
*Forward model error [%].*
- double [err\\_noise](#) [ND]  
*Noise error [ $W/(m^2 \text{ sr cm}^{-1})$ ].*
- double [err\\_press](#)  
*Pressure error [%].*
- double [err\\_press\\_cz](#)  
*Vertical correlation length for pressure error [km].*
- double [err\\_press\\_ch](#)  
*Horizontal correlation length for pressure error [km].*
- double [err\\_temp](#)  
*Temperature error [K].*
- double [err\\_temp\\_cz](#)  
*Vertical correlation length for temperature error [km].*
- double [err\\_temp\\_ch](#)  
*Horizontal correlation length for temperature error [km].*
- double [err\\_q](#) [NG]  
*Volume mixing ratio error [%].*
- double [err\\_q\\_cz](#) [NG]  
*Vertical correlation length for volume mixing ratio error [km].*
- double [err\\_q\\_ch](#) [NG]  
*Horizontal correlation length for volume mixing ratio error [km].*
- double [err\\_k](#) [NW]  
*Extinction error [1/km].*

- double `err_k_cz` [NW]  
*Vertical correlation length for extinction error [km].*
- double `err_k_ch` [NW]  
*Horizontal correlation length for extinction error [km].*
- double `err_clz`  
*Cloud height error [km].*
- double `err_cldz`  
*Cloud depth error [km].*
- double `err_clk` [NCL]  
*Cloud extinction error [1/km].*
- double `err_sfz`  
*Surface height error [km].*
- double `err_sfp`  
*Surface pressure error [hPa].*
- double `err_sft`  
*Surface temperature error [K].*
- double `err_steps` [NSF]  
*Surface emissivity error.*

#### 4.5.1 Detailed Description

Retrieval control parameters.

Definition at line 32 of file [retrieval.c](#).

#### 4.5.2 Field Documentation

##### 4.5.2.1 `dir` `char ret_t::dir[LEN]`

Working directory.

Definition at line 35 of file [retrieval.c](#).

##### 4.5.2.2 `kernel_recomp` `int ret_t::kernel_recomp`

Re-computation of kernel matrix (number of iterations).

Definition at line 38 of file [retrieval.c](#).

**4.5.2.3 `conv_itmax`** `int ret_t::conv_itmax`

Maximum number of iterations.

Definition at line 41 of file [retrieval.c](#).

**4.5.2.4 `conv_dmin`** `double ret_t::conv_dmin`

Minimum normalized step size in state space.

Definition at line 44 of file [retrieval.c](#).

**4.5.2.5 `err_ana`** `int ret_t::err_ana`

Carry out error analysis (0=no, 1=yes).

Definition at line 47 of file [retrieval.c](#).

**4.5.2.6 `err_formod`** `double ret_t::err_formod[ND]`

Forward model error [%].

Definition at line 50 of file [retrieval.c](#).

**4.5.2.7 `err_noise`** `double ret_t::err_noise[ND]`

Noise error [ $W/(m^2 \text{ sr cm}^{-1})$ ].

Definition at line 53 of file [retrieval.c](#).

**4.5.2.8 `err_press`** `double ret_t::err_press`

Pressure error [%].

Definition at line 56 of file [retrieval.c](#).

**4.5.2.9 err\_press\_cz** `double ret_t::err_press_cz`

Vertical correlation length for pressure error [km].

Definition at line 59 of file [retrieval.c](#).

**4.5.2.10 err\_press\_ch** `double ret_t::err_press_ch`

Horizontal correlation length for pressure error [km].

Definition at line 62 of file [retrieval.c](#).

**4.5.2.11 err\_temp** `double ret_t::err_temp`

Temperature error [K].

Definition at line 65 of file [retrieval.c](#).

**4.5.2.12 err\_temp\_cz** `double ret_t::err_temp_cz`

Vertical correlation length for temperature error [km].

Definition at line 68 of file [retrieval.c](#).

**4.5.2.13 err\_temp\_ch** `double ret_t::err_temp_ch`

Horizontal correlation length for temperature error [km].

Definition at line 71 of file [retrieval.c](#).

**4.5.2.14 err\_q** `double ret_t::err_q[NG]`

Volume mixing ratio error [%].

Definition at line 74 of file [retrieval.c](#).

**4.5.2.15 err\_q\_cz** `double ret_t::err_q_cz[NG]`

Vertical correlation length for volume mixing ratio error [km].

Definition at line 77 of file [retrieval.c](#).

**4.5.2.16 err\_q\_ch** `double ret_t::err_q_ch[NG]`

Horizontal correlation length for volume mixing ratio error [km].

Definition at line 80 of file [retrieval.c](#).

**4.5.2.17 err\_k** `double ret_t::err_k[NW]`

Extinction error [1/km].

Definition at line 83 of file [retrieval.c](#).

**4.5.2.18 err\_k\_cz** `double ret_t::err_k_cz[NW]`

Vertical correlation length for extinction error [km].

Definition at line 86 of file [retrieval.c](#).

**4.5.2.19 err\_k\_ch** `double ret_t::err_k_ch[NW]`

Horizontal correlation length for extinction error [km].

Definition at line 89 of file [retrieval.c](#).

**4.5.2.20 err\_clz** `double ret_t::err_clz`

Cloud height error [km].

Definition at line 92 of file [retrieval.c](#).

**4.5.2.21 err\_cldz** `double ret_t::err_cldz`

Cloud depth error [km].

Definition at line 95 of file [retrieval.c](#).

**4.5.2.22 err\_clk** `double ret_t::err_clk[NCL]`

Cloud extinction error [1/km].

Definition at line 98 of file [retrieval.c](#).

**4.5.2.23 err\_sfz** `double ret_t::err_sfz`

Surface height error [km].

Definition at line 101 of file [retrieval.c](#).

**4.5.2.24 err\_sfp** `double ret_t::err_sfp`

Surface pressure error [hPa].

Definition at line 104 of file [retrieval.c](#).

**4.5.2.25 err\_sft** `double ret_t::err_sft`

Surface temperature error [K].

Definition at line 107 of file [retrieval.c](#).

**4.5.2.26 err\_sfeps** `double ret_t::err_sfeps[NSF]`

Surface emissivity error.

Definition at line 110 of file [retrieval.c](#).

The documentation for this struct was generated from the following file:

- [retrieval.c](#)

## 4.6 tbl\_t Struct Reference

Emissivity look-up tables.

```
#include <jurassic.h>
```

### Data Fields

- int [np](#) [[ND](#)][[NG](#)]  
*Number of pressure levels.*
- int [nt](#) [[ND](#)][[NG](#)][[TBLNP](#)]  
*Number of temperatures.*
- int [nu](#) [[ND](#)][[NG](#)][[TBLNP](#)][[TBLNT](#)]  
*Number of column densities.*
- double [p](#) [[ND](#)][[NG](#)][[TBLNP](#)]  
*Pressure [hPa].*
- double [t](#) [[ND](#)][[NG](#)][[TBLNP](#)][[TBLNT](#)]  
*Temperature [K].*
- float [u](#) [[ND](#)][[NG](#)][[TBLNP](#)][[TBLNT](#)][[TBLNU](#)]  
*Column density [molecules/cm<sup>2</sup>].*
- float [eps](#) [[ND](#)][[NG](#)][[TBLNP](#)][[TBLNT](#)][[TBLNU](#)]  
*Emissivity.*
- double [st](#) [[TBLNS](#)]  
*Source function temperature [K].*
- double [sr](#) [[TBLNS](#)][[ND](#)]  
*Source function radiance [W/(m<sup>2</sup> sr cm<sup>-1</sup>)].*

### 4.6.1 Detailed Description

Emissivity look-up tables.

Definition at line [624](#) of file [jurassic.h](#).

### 4.6.2 Field Documentation

#### 4.6.2.1 np int tbl\_t::np[ND][NG]

Number of pressure levels.

Definition at line [627](#) of file [jurassic.h](#).



**4.6.2.2 nt** `int tbl_t::nt[ND][NG][TBLNP]`

Number of temperatures.

Definition at line 630 of file [jurassic.h](#).

**4.6.2.3 nu** `int tbl_t::nu[ND][NG][TBLNP][TBLNT]`

Number of column densities.

Definition at line 633 of file [jurassic.h](#).

**4.6.2.4 p** `double tbl_t::p[ND][NG][TBLNP]`

Pressure [hPa].

Definition at line 636 of file [jurassic.h](#).

**4.6.2.5 t** `double tbl_t::t[ND][NG][TBLNP][TBLNT]`

Temperature [K].

Definition at line 639 of file [jurassic.h](#).

**4.6.2.6 u** `float tbl_t::u[ND][NG][TBLNP][TBLNT][TBLNU]`

Column density [molecules/cm<sup>2</sup>].

Definition at line 642 of file [jurassic.h](#).

**4.6.2.7 eps** `float tbl_t::eps[ND][NG][TBLNP][TBLNT][TBLNU]`

Emissivity.

Definition at line 645 of file [jurassic.h](#).

**4.6.2.8** **st** double tbl\_t::st[TBLNS]

Source function temperature [K].

Definition at line 648 of file [jurassic.h](#).

**4.6.2.9** **sr** double tbl\_t::sr[TBLNS][ND]

Source function radiance [W/(m<sup>2</sup> sr cm<sup>-1</sup>)].

Definition at line 651 of file [jurassic.h](#).

The documentation for this struct was generated from the following file:

- [jurassic.h](#)

## 5 File Documentation

### 5.1 brightness.c File Reference

Convert radiance to brightness temperature.

```
#include "jurassic.h"
```

#### Functions

- int [main](#) (int argc, char \*argv[])

#### 5.1.1 Detailed Description

Convert radiance to brightness temperature.

Definition in file [brightness.c](#).

#### 5.1.2 Function Documentation

```

5.1.2.1 main() int main (
                int argc,
                char * argv[] )

```

Definition at line 27 of file [brightness.c](#).

```

00029     {
00030
00031     /* Check arguments... */
00032     if (argc != 3 && argc != 7)
00033         ERRMSG
00034         ("Give parameters: [ <rad> <nu> | "
00035          " <rad0> <rad1> <drad> <nu0> <nu1> <dnu> ]");
00036
00037     /* Calculate single value... */
00038     if (argc == 3) {
00039
00040         /* Read arguments... */
00041         double rad = atof(argv[1]);
00042         double nu = atof(argv[2]);
00043
00044         /* Compute brightness temperature... */
00045         printf("%.10g\n", brightness(rad, nu));
00046     }
00047
00048
00049     /* Calculate table... */
00050     else if (argc == 7) {
00051
00052         /* Read arguments... */
00053         double rad0 = atof(argv[1]);
00054         double rad1 = atof(argv[2]);
00055         double drad = atof(argv[3]);
00056         double nu0 = atof(argv[4]);
00057         double nu1 = atof(argv[5]);
00058         double dnu = atof(argv[6]);
00059
00060         /* Write header... */
00061         printf("# $1 = radiance [W/(m^2 sr cm^-1)]\n"
00062               "# $2 = wavenumber [cm^-1]\n"
00063               "# $3 = brightness temperature [K]\n");
00064
00065         /* Compute brightness temperature... */
00066         for (double rad = rad0; rad <= rad1; rad += drad) {
00067             printf("\n");
00068             for (double nu = nu0; nu <= nu1; nu += dnu)
00069                 printf("%.10g %.4f %.10g\n", rad, nu, brightness(rad, nu));
00070         }
00071     }
00072
00073     return EXIT_SUCCESS;
00074 }

```

Here is the call graph for this function:



## 5.2 brightness.c

```

00001 /*
00002  This file is part of JURASSIC.
00003
00004  JURASSIC is free software: you can redistribute it and/or modify
00005  it under the terms of the GNU General Public License as published by
00006  the Free Software Foundation, either version 3 of the License, or
00007  (at your option) any later version.
00008
00009  JURASSIC is distributed in the hope that it will be useful,
00010  but WITHOUT ANY WARRANTY; without even the implied warranty of

```

```

00011  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012  GNU General Public License for more details.
00013
00014  You should have received a copy of the GNU General Public License
00015  along with JURASSIC.  If not, see <http://www.gnu.org/licenses/>.
00016
00017  Copyright (C) 2003–2021 Forschungszentrum Juelich GmbH
00018  */
00019
00025  #include "jurassic.h"
00026
00027  int main(
00028      int argc,
00029      char *argv[]) {
00030
00031      /* Check arguments... */
00032      if (argc != 3 && argc != 7)
00033          ERRMSG
00034              ("Give parameters: [ <rad> <nu> | "
00035               " <rad0> <rad1> <drad> <nu0> <nu1> <dnu> ]");
00036
00037      /* Calculate single value... */
00038      if (argc == 3) {
00039
00040          /* Read arguments... */
00041          double rad = atof(argv[1]);
00042          double nu = atof(argv[2]);
00043
00044          /* Compute brightness temperature... */
00045          printf("%.10g\n", brightness(rad, nu));
00046      }
00047
00048      /* Calculate table... */
00049      else if (argc == 7) {
00050
00051          /* Read arguments... */
00052          double rad0 = atof(argv[1]);
00053          double rad1 = atof(argv[2]);
00054          double drad = atof(argv[3]);
00055          double nu0 = atof(argv[4]);
00056          double nu1 = atof(argv[5]);
00057          double dnu = atof(argv[6]);
00058
00059          /* Write header... */
00060          printf("# $1 = radiance [W/(m^2 sr cm^-1)]\n"
00061                 "# $2 = wavenumber [cm^-1]\n"
00062                 "# $3 = brightness temperature [K]\n");
00063
00064          /* Compute brightness temperature... */
00065          for (double rad = rad0; rad <= rad1; rad += drad) {
00066              printf("\n");
00067              for (double nu = nu0; nu <= nu1; nu += dnu)
00068                  printf("%.10g %.4f %.10g\n", rad, nu, brightness(rad, nu));
00069          }
00070      }
00071  }
00072
00073  return EXIT_SUCCESS;
00074 }

```

## 5.3 climatology.c File Reference

Prepare atmospheric data file from climatological data.

```
#include "jurassic.h"
```

### Functions

- `int main` (int argc, char \*argv[])

#### 5.3.1 Detailed Description

Prepare atmospheric data file from climatological data.

Definition in file [climatology.c](#).

## 5.3.2 Function Documentation

**5.3.2.1 main()** `int main (`  
`int argc,`  
`char * argv[] )`

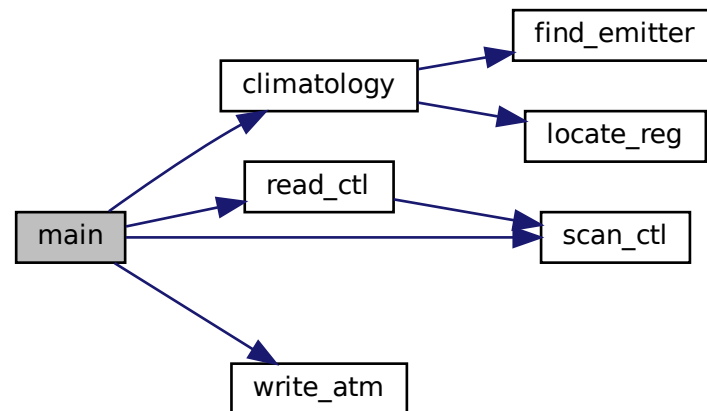
Definition at line 27 of file [climatology.c](#).

```

00029     {
00030
00031     static atm_t atm;
00032     static ctl_t ctl;
00033
00034     double clk[NCL], sfeps[NSF];
00035
00036     /* Check arguments... */
00037     if (argc < 3)
00038         ERRMSG("Give parameters: <ctl> <atm>");
00039
00040     /* Read control parameters... */
00041     read_ctl(argc, argv, &ctl);
00042     double t0 = scan_ctl(argc, argv, "T0", -1, "0", NULL);
00043     double t1 = scan_ctl(argc, argv, "T1", -1, "0", NULL);
00044     double dt = scan_ctl(argc, argv, "DT", -1, "1", NULL);
00045     double z0 = scan_ctl(argc, argv, "Z0", -1, "0", NULL);
00046     double z1 = scan_ctl(argc, argv, "Z1", -1, "90", NULL);
00047     double dz = scan_ctl(argc, argv, "DZ", -1, "1", NULL);
00048     double clz = scan_ctl(argc, argv, "CLZ", -1, "0", NULL);
00049     double cldz = scan_ctl(argc, argv, "CLDZ", -1, "0", NULL);
00050     for (int icl = 0; icl < ctl.ncl; icl++)
00051         clk[icl] = scan_ctl(argc, argv, "CLK", icl, "0", NULL);
00052     double sfz = scan_ctl(argc, argv, "SFZ", -1, "0", NULL);
00053     double sfp = scan_ctl(argc, argv, "SFP", -1, "0", NULL);
00054     double sft = scan_ctl(argc, argv, "SFT", -1, "0", NULL);
00055     for (int isf = 0; isf < ctl.nsf; isf++)
00056         sfeps[isf] = scan_ctl(argc, argv, "SFEPS", isf, "1", NULL);
00057
00058     /* Set atmospheric grid... */
00059     for (double t = t0; t <= t1; t += dt)
00060         for (double z = z0; z <= z1; z += dz) {
00061             atm.time[atm.np] = t;
00062             atm.z[atm.np] = z;
00063             if ((++atm.np) >= NP)
00064                 ERRMSG("Too many atmospheric grid points!");
00065         }
00066
00067     /* Interpolate climatological data... */
00068     climatology(&ctl, &atm);
00069
00070     /* Set cloud layer... */
00071     atm.clz = clz;
00072     atm.cldz = cldz;
00073     for (int icl = 0; icl < ctl.ncl; icl++)
00074         atm.clk[icl] = clk[icl];
00075
00076     /* Set surface layer... */
00077     atm.sfz = sfz;
00078     atm.sfp = sfp;
00079     atm.sft = sft;
00080     for (int isf = 0; isf < ctl.nsf; isf++)
00081         atm.sfeps[isf] = sfeps[isf];
00082
00083     /* Write data to disk... */
00084     write_atm(NULL, argv[2], &ctl, &atm);
00085
00086     return EXIT_SUCCESS;
00087 }

```

Here is the call graph for this function:



## 5.4 climatology.c

```

00001 /*
00002  This file is part of JURASSIC.
00003
00004  JURASSIC is free software: you can redistribute it and/or modify
00005  it under the terms of the GNU General Public License as published by
00006  the Free Software Foundation, either version 3 of the License, or
00007  (at your option) any later version.
00008
00009  JURASSIC is distributed in the hope that it will be useful,
00010  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  GNU General Public License for more details.
00013
00014  You should have received a copy of the GNU General Public License
00015  along with JURASSIC. If not, see <http://www.gnu.org/licenses/>.
00016
00017  Copyright (C) 2003-2021 Forschungszentrum Juelich GmbH
00018 */
00019
00025 #include "jurassic.h"
00026
00027 int main(
00028     int argc,
00029     char *argv[]) {
00030
00031     static atm_t atm;
00032     static ctl_t ctl;
00033
00034     double clk[NCL], sfeps[NSF];
00035
00036     /* Check arguments... */
00037     if (argc < 3)
00038         ERRMSG("Give parameters: <ctl> <atm>");
00039
00040     /* Read control parameters... */
00041     read_ctl(argc, argv, &ctl);
00042     double t0 = scan_ctl(argc, argv, "T0", -1, "0", NULL);
00043     double t1 = scan_ctl(argc, argv, "T1", -1, "0", NULL);
00044     double dt = scan_ctl(argc, argv, "DT", -1, "1", NULL);
00045     double z0 = scan_ctl(argc, argv, "Z0", -1, "0", NULL);
00046     double z1 = scan_ctl(argc, argv, "Z1", -1, "90", NULL);
00047     double dz = scan_ctl(argc, argv, "DZ", -1, "1", NULL);
00048     double clz = scan_ctl(argc, argv, "CLZ", -1, "0", NULL);
00049     double cldz = scan_ctl(argc, argv, "CLDZ", -1, "0", NULL);
00050     for (int icl = 0; icl < ctl.ncl; icl++)
00051         clk[icl] = scan_ctl(argc, argv, "CLK", icl, "0", NULL);
00052     double sfz = scan_ctl(argc, argv, "SFZ", -1, "0", NULL);
00053     double sfp = scan_ctl(argc, argv, "SFP", -1, "0", NULL);
  
```

```

00054 double sft = scan_ctl(argc, argv, "SFT", -1, "0", NULL);
00055 for (int isf = 0; isf < ctl.nsf; isf++)
00056     sfeps[isf] = scan_ctl(argc, argv, "SFEPS", isf, "1", NULL);
00057
00058 /* Set atmospheric grid... */
00059 for (double t = t0; t <= t1; t += dt)
00060     for (double z = z0; z <= z1; z += dz) {
00061         atm.time[atm.np] = t;
00062         atm.z[atm.np] = z;
00063         if ((++atm.np) >= NP)
00064             ERRMSG("Too many atmospheric grid points!");
00065     }
00066
00067 /* Interpolate climatological data... */
00068 climatology(&ctl, &atm);
00069
00070 /* Set cloud layer... */
00071 atm.clz = clz;
00072 atm.cldz = cldz;
00073 for (int icl = 0; icl < ctl.ncl; icl++)
00074     atm.clk[icl] = clk[icl];
00075
00076 /* Set surface layer... */
00077 atm.sfz = sfz;
00078 atm.sfp = sfp;
00079 atm.sft = sft;
00080 for (int isf = 0; isf < ctl.nsf; isf++)
00081     atm.sfeps[isf] = sfeps[isf];
00082
00083 /* Write data to disk... */
00084 write_atm(NULL, argv[2], &ctl, &atm);
00085
00086 return EXIT_SUCCESS;
00087 }

```

## 5.5 filter.c File Reference

Create radiometric filter functions.

```
#include "jurassic.h"
```

### Functions

- double [ails](#) (int apo, double opl, double dnu)  
*Compute apodized instrument line shape.*
- int [main](#) (int argc, char \*argv[])

#### 5.5.1 Detailed Description

Create radiometric filter functions.

Definition in file [filter.c](#).

#### 5.5.2 Function Documentation

**5.5.2.1 ails()** double ails (  
int apo,  
double opl,  
double dnu )

Compute apodized instrument line shape.

Definition at line 120 of file [filter.c](#).

```
00123     {
00124
00125     double a, a2, a4, a6, a8, cosa, q0, q2, q4, sinca;
00126
00127     /* Auxiliary quantities... */
00128     a = 2 * M_PI * dnu * opl;
00129     a2 = a * a;
00130     a4 = a2 * a2;
00131     a6 = a4 * a2;
00132     a8 = a4 * a4;
00133
00134     /* Sinc function... */
00135     if (apo == 0) {
00136         if (fabs(a) < 0.7)
00137             return 1 - a2 / 6 + a4 / 120 - a6 / 5040 + a8 / 362880;
00138         else
00139             return sin(a) / a;
00140     }
00141
00142     /* Norton-Beer strong apodization... */
00143     else if (apo == 1) {
00144         if (fabs(a) < 0.7) {
00145             q0 = 1 - a2 / 6 + a4 / 120 - a6 / 5040 + a8 / 362880;
00146             q2 = 1 - a2 / 14 + a4 / 504 - a6 / 33264 + a8 / 3459456;
00147             q4 = 1 - a2 / 22 + a4 / 1144 - a6 / 102960 + a8 / 14002560;
00148         } else {
00149             sinca = sin(a) / a;
00150             cosa = cos(a);
00151             q0 = sinca;
00152             q2 = -15 / a2 * ((1 - 3 / a2) * sinca + (3 / a2) * cosa);
00153             q4 =
00154                 945 / a4 * ((1 - 45 / a2 + 105 / a4) * sinca +
00155                     5 / a2 * (2 - 21 / a2) * cosa);
00156         }
00157         return 0.045335 * q0 + 0.554883 * q2 * 8. / 15. +
00158             0.399782 * q4 * 384. / 945.;
00159     }
00160
00161     /* Error message.... */
00162     else
00163         ERRMSG("Unknown apodization!");
00164 }
```

**5.5.2.2 main()** int main (  
int argc,  
char \* argv[] )

Definition at line 41 of file [filter.c](#).

```
00043     {
00044
00045     static ctl_t ctl;
00046
00047     static double ff[NSHAPE], fnu[NSHAPE];
00048
00049     double fsum = 0.0;
00050
00051     int fn = 0;
00052
00053     /* Write info... */
00054     if (argc < 3)
00055         ERRMSG("Give parameters: <ctl> <filter>");
00056
00057     /* Read control parameters... */
00058     read_ctl(argc, argv, &ctl);
00059     int type = (int) scan_ctl(argc, argv, "FILTER_TYPE", -1, "1", NULL);
00060     double opd = scan_ctl(argc, argv, "FILTER_OPD", -1, "10.0", NULL);
00061     double fwhm = scan_ctl(argc, argv, "FILTER_FWHM", -1, "1.0", NULL);
00062     double center = scan_ctl(argc, argv, "FILTER_CENTER", -1, "1000.0", NULL);
```

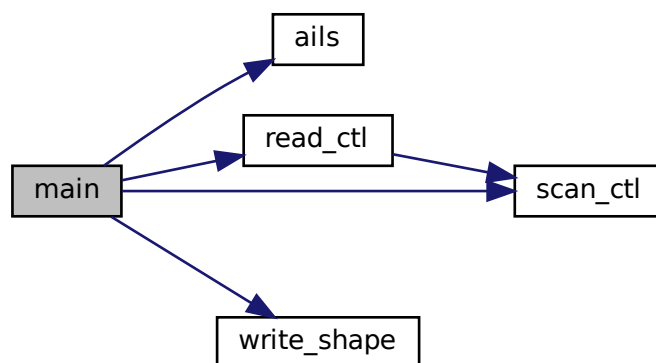


```

00063 double width = scan_ctl(argc, argv, "FILTER_WIDTH", -1, "2.1", NULL);
00064 double samp = scan_ctl(argc, argv, "FILTER_SAMP", -1, "0.0005", NULL);
00065
00066 /* Compute filter function... */
00067 for (double nu = center - 0.5 * width;
00068      nu <= center + 0.5 * width; nu += samp) {
00069
00070     /* Set frequency... */
00071     fnu[fn] = nu;
00072
00073     /* Boxcar... */
00074     if (type == 0)
00075         ff[fn] = (fabs(nu - center) <= 0.5 * fwhm ? 1.0 : 0.0);
00076
00077     /* Triangle... */
00078     else if (type == 1) {
00079         ff[fn] = 1.0 - fabs(nu - center) / fwhm;
00080         ff[fn] = GSL_MAX(ff[fn], 0.0);
00081     }
00082
00083     /* Gaussian... */
00084     else if (type == 2) {
00085         double sigma = fwhm / 2.355;
00086         ff[fn] = exp(-0.5 * POW2((nu - center) / sigma));
00087     }
00088
00089     /* Sinc function... */
00090     else if (type == 3)
00091         ff[fn] = ails(0, opd, nu - center);
00092
00093     /* Norton-Beer strong apodization... */
00094     else if (type == 4)
00095         ff[fn] = ails(1, opd, nu - center);
00096
00097     /* Error message... */
00098     else
00099         ERRMSG("Filter function type unknown!");
00100
00101     /* Count spectral grid points... */
00102     if ((++fn) >= NSHAPE)
00103         ERRMSG("Too many filter function data points!");
00104 }
00105
00106 /* Normalize filter function... */
00107 for (int i = 0; i < fn; i++)
00108     fsum += ff[i];
00109 for (int i = 0; i < fn; i++)
00110     ff[i] /= (fsum * samp);
00111
00112 /* Write to file... */
00113 write_shape(argv[2], fnu, ff, fn);
00114
00115 return (EXIT_SUCCESS);
00116 }

```

Here is the call graph for this function:



## 5.6 filter.c

```

00001 /*
00002   This file is part of JURASSIC.
00003
00004   JURASSIC is free software: you can redistribute it and/or modify
00005   it under the terms of the GNU General Public License as published by
00006   the Free Software Foundation, either version 3 of the License, or
00007   (at your option) any later version.
00008
00009   JURASSIC is distributed in the hope that it will be useful,
00010   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012   GNU General Public License for more details.
00013
00014   You should have received a copy of the GNU General Public License
00015   along with JURASSIC. If not, see <http://www.gnu.org/licenses/>.
00016
00017   Copyright (C) 2003-2021 Forschungszentrum Juelich GmbH
00018 */
00019
00025 #include "jurassic.h"
00026
00027 /* -----
00028   Functions...
00029   ----- */
00030
00032 double ails(
00033     int apo,
00034     double opl,
00035     double dnu);
00036
00037 /* -----
00038   Main...
00039   ----- */
00040
00041 int main(
00042     int argc,
00043     char *argv[]) {
00044
00045     static ctl_t ctl;
00046
00047     static double ff[NSHAPE], fnu[NSHAPE];
00048
00049     double fsum = 0.0;
00050
00051     int fn = 0;
00052
00053     /* Write info... */
00054     if (argc < 3)
00055         ERRMSG("Give parameters: <ctl> <filter>");
00056
00057     /* Read control parameters... */
00058     read_ctl(argc, argv, &ctl);
00059     int type = (int) scan_ctl(argc, argv, "FILTER_TYPE", -1, "1", NULL);
00060     double opd = scan_ctl(argc, argv, "FILTER_OPD", -1, "10.0", NULL);
00061     double fwhm = scan_ctl(argc, argv, "FILTER_FWHM", -1, "1.0", NULL);
00062     double center = scan_ctl(argc, argv, "FILTER_CENTER", -1, "1000.0", NULL);
00063     double width = scan_ctl(argc, argv, "FILTER_WIDTH", -1, "2.1", NULL);
00064     double samp = scan_ctl(argc, argv, "FILTER_SAMP", -1, "0.0005", NULL);
00065
00066     /* Compute filter function... */
00067     for (double nu = center - 0.5 * width;
00068         nu <= center + 0.5 * width; nu += samp) {
00069
00070         /* Set frequency... */
00071         fnu[fn] = nu;
00072
00073         /* Boxcar... */
00074         if (type == 0)
00075             ff[fn] = (fabs(nu - center) <= 0.5 * fwhm ? 1.0 : 0.0);
00076
00077         /* Triangle... */
00078         else if (type == 1) {
00079             ff[fn] = 1.0 - fabs(nu - center) / fwhm;
00080             ff[fn] = GSL_MAX(ff[fn], 0.0);
00081         }
00082
00083         /* Gaussian... */
00084         else if (type == 2) {
00085             double sigma = fwhm / 2.355;
00086             ff[fn] = exp(-0.5 * POW2((nu - center) / sigma));
00087         }
00088
00089         /* Sinc function... */
00090         else if (type == 3)
00091             ff[fn] = ails(0, opd, nu - center);

```

```

00092
00093     /* Norton-Beer strong apodization... */
00094     else if (type == 4)
00095         ff[fn] = ails(1, opd, nu - center);
00096
00097     /* Error message... */
00098     else
00099         ERRMSG("Filter function type unknown!");
00100
00101     /* Count spectral grid points... */
00102     if ((++fn) >= NSHAPE)
00103         ERRMSG("Too many filter function data points!");
00104 }
00105
00106 /* Normalize filter function... */
00107 for (int i = 0; i < fn; i++)
00108     fsum += ff[i];
00109 for (int i = 0; i < fn; i++)
00110     ff[i] /= (fsum * samp);
00111
00112 /* Write to file... */
00113 write_shape(argv[2], fnu, ff, fn);
00114
00115 return (EXIT_SUCCESS);
00116 }
00117
00118 /*****
00119
00120 double ails(
00121     int apo,
00122     double opl,
00123     double dnu) {
00124
00125     double a, a2, a4, a6, a8, cosa, q0, q2, q4, sinca;
00126
00127     /* Auxiliary quantities... */
00128     a = 2 * M_PI * dnu * opl;
00129     a2 = a * a;
00130     a4 = a2 * a2;
00131     a6 = a4 * a2;
00132     a8 = a4 * a4;
00133
00134     /* Sinc function... */
00135     if (apo == 0) {
00136         if (fabs(a) < 0.7)
00137             return 1 - a2 / 6 + a4 / 120 - a6 / 5040 + a8 / 362880;
00138         else
00139             return sin(a) / a;
00140     }
00141
00142     /* Norton-Beer strong apodization... */
00143     else if (apo == 1) {
00144         if (fabs(a) < 0.7) {
00145             q0 = 1 - a2 / 6 + a4 / 120 - a6 / 5040 + a8 / 362880;
00146             q2 = 1 - a2 / 14 + a4 / 504 - a6 / 33264 + a8 / 3459456;
00147             q4 = 1 - a2 / 22 + a4 / 1144 - a6 / 102960 + a8 / 14002560;
00148         } else {
00149             sinca = sin(a) / a;
00150             cosa = cos(a);
00151             q0 = sinca;
00152             q2 = -15 / a2 * ((1 - 3 / a2) * sinca + (3 / a2) * cosa);
00153             q4 =
00154                 945 / a4 * ((1 - 45 / a2 + 105 / a4) * sinca +
00155                     5 / a2 * (2 - 21 / a2) * cosa);
00156         }
00157         return 0.045335 * q0 + 0.554883 * q2 * 8. / 15. +
00158             0.399782 * q4 * 384. / 945.;
00159     }
00160
00161     /* Error message.... */
00162     else
00163         ERRMSG("Unknown apodization!");
00164 }

```

## 5.7 formod.c File Reference

JURASSIC forward model.

```
#include "jurassic.h"
```

## Functions

- void [call\\_formod](#) ([ctl\\_t](#) \*ctl, const char \*wrkdir, const char \*obsfile, const char \*atmfile, const char \*radfile, const char \*task)  
*Perform forward model calculations in a single directory.*
- int [main](#) (int argc, char \*argv[])

### 5.7.1 Detailed Description

JURASSIC forward model.

Definition in file [formod.c](#).

### 5.7.2 Function Documentation

**5.7.2.1 [call\\_formod\(\)](#)** void [call\\_formod](#) (  
     [ctl\\_t](#) \* [ctl](#),  
     const char \* [wrkdir](#),  
     const char \* [obsfile](#),  
     const char \* [atmfile](#),  
     const char \* [radfile](#),  
     const char \* [task](#) )

Perform forward model calculations in a single directory.

Definition at line 122 of file [formod.c](#).

```
00128     {
00129
00130     static atm_t atm, atm2;
00131     static obs_t obs, obs2;
00132
00133     char filename[LEN];
00134
00135     /* Read observation geometry... */
00136     read_obs(wrkdir, obsfile, ctl, &obs);
00137
00138     /* Read atmospheric data... */
00139     read_atm(wrkdir, atmfile, ctl, &atm);
00140
00141     /* Compute multiple profiles... */
00142     if (task[0] == 'p' || task[0] == 'P') {
00143
00144         /* Loop over ray paths... */
00145         for (int ir = 0; ir < obs.nr; ir++) {
00146
00147             /* Get atmospheric data... */
00148             atm2.np = 0;
00149             for (int ip = 0; ip < atm.np; ip++)
00150                 if (atm.time[ip] == obs.time[ir]) {
00151                     atm2.time[atm2.np] = atm.time[ip];
00152                     atm2.z[atm2.np] = atm.z[ip];
00153                     atm2.lon[atm2.np] = atm.lon[ip];
00154                     atm2.lat[atm2.np] = atm.lat[ip];
00155                     atm2.p[atm2.np] = atm.p[ip];
00156                     atm2.t[atm2.np] = atm.t[ip];
00157                     for (int ig = 0; ig < ctl->ng; ig++)
00158                         atm2.q[ig][atm2.np] = atm.q[ig][ip];
00159                     for (int iw = 0; iw < ctl->nw; iw++)
00160                         atm2.k[iw][atm2.np] = atm.k[iw][ip];
00161                     atm2.np++;
00162                 }
00163
00164             /* Get observation data... */
00165             obs2.nr = 1;
```

```

00166     obs2.time[0] = obs.time[ir];
00167     obs2.vpz[0] = obs.vpz[ir];
00168     obs2.vplon[0] = obs.vplon[ir];
00169     obs2.vplat[0] = obs.vplat[ir];
00170     obs2.obsz[0] = obs.obsz[ir];
00171     obs2.obslon[0] = obs.obslon[ir];
00172     obs2.obslat[0] = obs.obslat[ir];
00173
00174     /* Check number of data points... */
00175     if (atm2.np > 0) {
00176
00177         /* Call forward model... */
00178         formod(ctl, &atm2, &obs2);
00179
00180         /* Save radiance data... */
00181         for (int id = 0; id < ctl->nd; id++) {
00182             obs.rad[id][ir] = obs2.rad[id][0];
00183             obs.tau[id][ir] = obs2.tau[id][0];
00184         }
00185     }
00186 }
00187
00188 /* Write radiance data... */
00189 write_obs(wrkdir, radfile, ctl, &obs);
00190 }
00191
00192 /* Compute single profile... */
00193 else {
00194
00195     /* Call forward model... */
00196     formod(ctl, &atm, &obs);
00197
00198     /* Save radiance data... */
00199     write_obs(wrkdir, radfile, ctl, &obs);
00200
00201     /* Compute contributions... */
00202     if (task[0] == 'c' || task[0] == 'C') {
00203
00204         /* Switch off continua... */
00205         ctl->ctm_co2 = 0;
00206         ctl->ctm_h2o = 0;
00207         ctl->ctm_n2 = 0;
00208         ctl->ctm_o2 = 0;
00209
00210         /* Loop over emitters... */
00211         for (int ig = 0; ig < ctl->ng; ig++) {
00212
00213             /* Copy atmospheric data... */
00214             copy_atm(ctl, &atm2, &atm, 0);
00215
00216             /* Set extinction to zero... */
00217             for (int iw = 0; iw < ctl->nw; iw++)
00218                 for (int ip = 0; ip < atm2.np; ip++)
00219                     atm2.k[iw][ip] = 0;
00220
00221             /* Set volume mixing ratios to zero... */
00222             for (int ig2 = 0; ig2 < ctl->ng; ig2++)
00223                 if (ig2 != ig)
00224                     for (int ip = 0; ip < atm2.np; ip++)
00225                         atm2.q[ig2][ip] = 0;
00226
00227             /* Call forward model... */
00228             formod(ctl, &atm2, &obs);
00229
00230             /* Save radiance data... */
00231             sprintf(filename, "%s.%s", radfile, ctl->emitter[ig]);
00232             write_obs(wrkdir, filename, ctl, &obs);
00233         }
00234
00235         /* Copy atmospheric data... */
00236         copy_atm(ctl, &atm2, &atm, 0);
00237
00238         /* Set volume mixing ratios to zero... */
00239         for (int ig = 0; ig < ctl->ng; ig++)
00240             for (int ip = 0; ip < atm2.np; ip++)
00241                 atm2.q[ig][ip] = 0;
00242
00243         /* Call forward model... */
00244         formod(ctl, &atm2, &obs);
00245
00246         /* Save radiance data... */
00247         sprintf(filename, "%s.EXTINCT", radfile);
00248         write_obs(wrkdir, filename, ctl, &obs);
00249     }
00250
00251     /* Measure CPU-time... */
00252     if (task[0] == 't' || task[0] == 'T') {

```

```

00253
00254     /* Init... */
00255     double t_min, t_max, t_mean = 0, t_sigma = 0;
00256     int n = 0;
00257
00258     /* Initialize random number generator... */
00259     gsl_rng_env_setup();
00260     gsl_rng *rng = gsl_rng_alloc(gsl_rng_default);
00261
00262     /* Loop over profiles... */
00263     do {
00264
00265         /* Create random atmosphere... */
00266         copy_atm(ctl, &atm2, &atm, 0);
00267         double dtemp = 40. * (gsl_rng_uniform(rng) - 0.5);
00268         double dpress = 1. - 0.1 * gsl_rng_uniform(rng);
00269         double dq[NG];
00270         for (int ig = 0; ig < ctl->ng; ig++)
00271             dq[ig] = 0.8 + 0.4 * gsl_rng_uniform(rng);
00272         for (int ip = 0; ip < atm2.np; ip++) {
00273             atm.t[ip] += dtemp;
00274             atm.p[ip] *= dpress;
00275             for (int ig = 0; ig < ctl->ng; ig++)
00276                 atm.q[ig][ip] *= dq[ig];
00277         }
00278
00279         /* Measure runtime... */
00280         double t0 = omp_get_wtime();
00281         formod(ctl, &atm2, &obs);
00282         double dt = omp_get_wtime() - t0;
00283
00284         /* Get runtime statistics... */
00285         t_mean += (dt);
00286         t_sigma += POW2(dt);
00287         if (n == 0 || dt < t_min)
00288             t_min = dt;
00289         if (n == 0 || dt > t_max)
00290             t_max = dt;
00291         n++;
00292     } while (t_mean < 10.0);
00293
00294     /* Write results... */
00295     t_mean /= (double) n;
00296     t_sigma = sqrt(t_sigma / (double) n - POW2(t_mean));
00297     printf("RUNTIME_MEAN = %g s\n", t_mean);
00298     printf("RUNTIME_SIGMA = %g s\n", t_sigma);
00299     printf("RUNTIME_MIN = %g s\n", t_min);
00300     printf("RUNTIME_MAX = %g s\n", t_max);
00301     printf("RAYS_PER_SECOND = %g", (double) obs.nr / t_mean);
00302 }
00303
00304
00305     /* Analyze effect of step size... */
00306     if (task[0] == 's' || task[0] == 'S') {
00307
00308         /* Reference run... */
00309         ctl->rayds = 0.1;
00310         ctl->raydz = 0.01;
00311         formod(ctl, &atm, &obs);
00312         copy_obs(ctl, &obs2, &obs, 0);
00313
00314         /* Loop over step size... */
00315         for (double dz = 0.01; dz <= 2; dz *= 1.1) {
00316             printf("STEP SIZE: %g\n", dz);
00317             for (double ds = 0.1; ds <= 50; ds *= 1.1) {
00318
00319                 /* Set step size... */
00320                 ctl->rayds = ds;
00321                 ctl->raydz = dz;
00322
00323                 /* Measure runtime... */
00324                 double t0 = omp_get_wtime();
00325                 formod(ctl, &atm, &obs);
00326                 double dt = omp_get_wtime() - t0;
00327
00328                 /* Get differences... */
00329                 double mean[ND], sigma[ND];
00330                 for (int id = 0; id < ctl->nd; id++) {
00331                     mean[id] = sigma[id] = 0;
00332                     int n = 0;
00333                     for (int ir = 0; ir < obs.nr; ir++) {
00334                         double err = 200. * (obs.rad[id][ir] - obs2.rad[id][ir])
00335                             / (obs.rad[id][ir] + obs2.rad[id][ir]);
00336                         mean[id] += err;
00337                         sigma[id] += POW2(err);
00338                         n++;
00339                     }

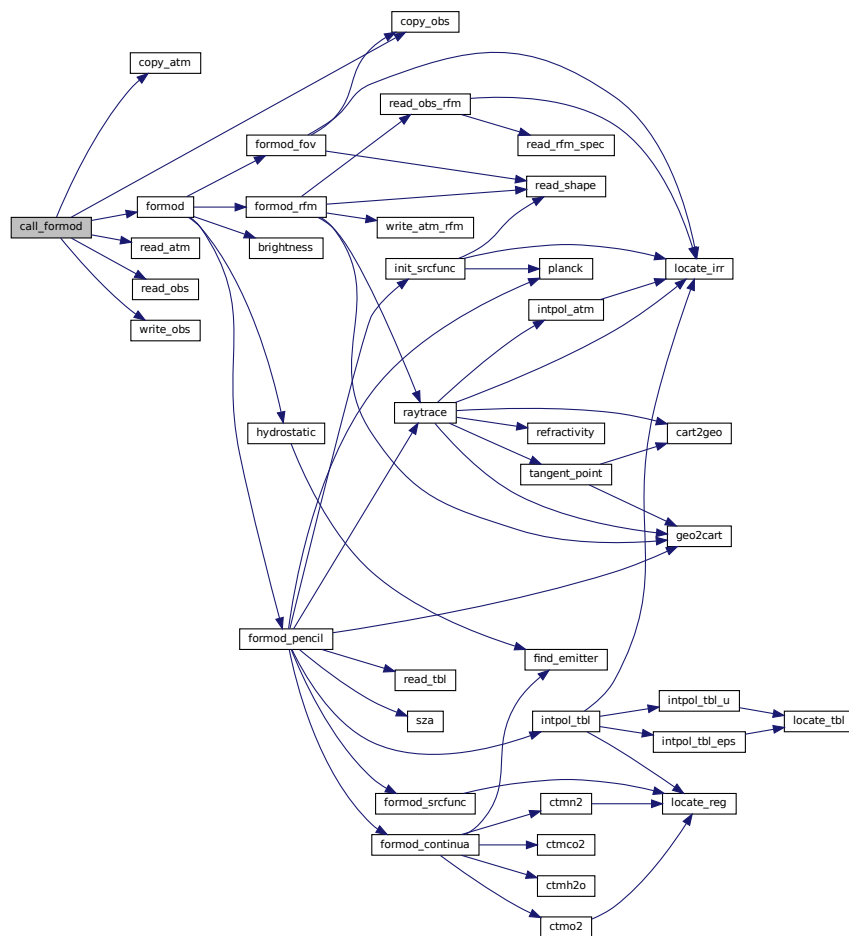
```

```

00340         mean[id] /= n;
00341         sigma[id] = sqrt(sigma[id] / n - POW2(mean[id]));
00342     }
00343
00344     /* Write results... */
00345     printf("STEPsize: %g %g %g", ds, dz, dt);
00346     for (int id = 0; id < ctl->nd; id++)
00347         printf(" %g %g", mean[id], sigma[id]);
00348     printf("\n");
00349 }
00350 }
00351 }
00352 }
00353 }

```

Here is the call graph for this function:



```

5.7.2.2 main() int main (
                int argc,
                char * argv[] )

```

Definition at line 47 of file [formod.c](#).

```

00049     {
00050
00051         static ctl_t ctl;
00052

```

```

00053  /* Check arguments... */
00054  if (argc < 5)
00055      ERRMSG("Give parameters: <ctl> <obs> <atm> <rad>");
00056
00057  /* Read control parameters... */
00058  read_ctl(argc, argv, &ctl);
00059
00060 #ifdef UNIFIED
00061
00062  static atm_t atm;
00063  static obs_t obs;
00064
00065  /* Read observation geometry... */
00066  read_obs(NULL, argv[2], &ctl, &obs);
00067
00068  /* Read atmospheric data... */
00069  read_atm(NULL, argv[3], &ctl, &atm);
00070
00071  /* Call forward model... */
00072  jur_unified_init(argc, argv);
00073  jur_unified_formod_multiple_packages(&atm, &obs, 1, NULL);
00074
00075  /* Save radiance data... */
00076  write_obs(NULL, argv[4], &ctl, &obs);
00077
00078 #else
00079
00080  FILE *in;
00081
00082  char dirlist[LEN], task[LEN], wrkdir[LEN];
00083
00084  /* Get task... */
00085  scan_ctl(argc, argv, "TASK", -1, "-", task);
00086
00087  /* Get dirlist... */
00088  scan_ctl(argc, argv, "DIRLIST", -1, "-", dirlist);
00089
00090  /* Single forward calculation... */
00091  if (dirlist[0] == '-')
00092      call_formod(&ctl, NULL, argv[2], argv[3], argv[4], task);
00093
00094  /* Work on directory list... */
00095  else {
00096
00097      /* Open directory list... */
00098      if (!(in = fopen(dirlist, "r")))
00099          ERRMSG("Cannot open directory list!");
00100
00101      /* Loop over directories... */
00102      while (fscanf(in, "%s", wrkdir) != EOF) {
00103
00104          /* Write info... */
00105          LOG(1, "\nWorking directory: %s", wrkdir);
00106
00107          /* Call forward model... */
00108          call_formod(&ctl, wrkdir, argv[2], argv[3], argv[4], task);
00109      }
00110
00111      /* Close dirlist... */
00112      fclose(in);
00113  }
00114
00115 #endif
00116
00117  return EXIT_SUCCESS;
00118 }

```





```

00045 ----- */
00046
00047 int main(
00048     int argc,
00049     char *argv[]) {
00050
00051     static ctl_t ctl;
00052
00053     /* Check arguments... */
00054     if (argc < 5)
00055         ERRMSG("Give parameters: <ctl> <obs> <atm> <rad>");
00056
00057     /* Read control parameters... */
00058     read_ctl(argc, argv, &ctl);
00059
00060 #ifdef UNIFIED
00061
00062     static atm_t atm;
00063     static obs_t obs;
00064
00065     /* Read observation geometry... */
00066     read_obs(NULL, argv[2], &ctl, &obs);
00067
00068     /* Read atmospheric data... */
00069     read_atm(NULL, argv[3], &ctl, &atm);
00070
00071     /* Call forward model... */
00072     jur_unified_init(argc, argv);
00073     jur_unified_formod_multiple_packages(&atm, &obs, 1, NULL);
00074
00075     /* Save radiance data... */
00076     write_obs(NULL, argv[4], &ctl, &obs);
00077
00078 #else
00079     FILE *in;
00080
00081     char dirlist[LEN], task[LEN], wrkdir[LEN];
00082
00083     /* Get task... */
00084     scan_ctl(argc, argv, "TASK", -1, "-", task);
00085
00086     /* Get dirlist... */
00087     scan_ctl(argc, argv, "DIRLIST", -1, "-", dirlist);
00088
00089     /* Single forward calculation... */
00090     if (dirlist[0] == '-')
00091         call_formod(&ctl, NULL, argv[2], argv[3], argv[4], task);
00092
00093     /* Work on directory list... */
00094     else {
00095
00096         /* Open directory list... */
00097         if (!(in = fopen(dirlist, "r")))
00098             ERRMSG("Cannot open directory list!");
00099
00100         /* Loop over directories... */
00101         while (fscanf(in, "%s", wrkdir) != EOF) {
00102
00103             /* Write info... */
00104             LOG(1, "\nWorking directory: %s", wrkdir);
00105
00106             /* Call forward model... */
00107             call_formod(&ctl, wrkdir, argv[2], argv[3], argv[4], task);
00108         }
00109
00110         /* Close dirlist... */
00111         fclose(in);
00112     }
00113
00114 #endif
00115
00116     return EXIT_SUCCESS;
00117 }
00118
00119 /*****
00120
00121 void call_formod(
00122     ctl_t *ctl,
00123     const char *wrkdir,
00124     const char *obsfile,
00125     const char *atmfile,
00126     const char *radfile,
00127     const char *task) {
00128
00129     static atm_t atm, atm2;
00130     static obs_t obs, obs2;

```

```

00132
00133 char filename[LEN];
00134
00135 /* Read observation geometry... */
00136 read_obs(wrkdir, obsfile, ctl, &obs);
00137
00138 /* Read atmospheric data... */
00139 read_atm(wrkdir, atmfile, ctl, &atm);
00140
00141 /* Compute multiple profiles... */
00142 if (task[0] == 'p' || task[0] == 'P') {
00143
00144     /* Loop over ray paths... */
00145     for (int ir = 0; ir < obs.nr; ir++) {
00146
00147         /* Get atmospheric data... */
00148         atm2.np = 0;
00149         for (int ip = 0; ip < atm.np; ip++)
00150             if (atm.time[ip] == obs.time[ir]) {
00151                 atm2.time[atm2.np] = atm.time[ip];
00152                 atm2.z[atm2.np] = atm.z[ip];
00153                 atm2.lon[atm2.np] = atm.lon[ip];
00154                 atm2.lat[atm2.np] = atm.lat[ip];
00155                 atm2.p[atm2.np] = atm.p[ip];
00156                 atm2.t[atm2.np] = atm.t[ip];
00157                 for (int ig = 0; ig < ctl->ng; ig++)
00158                     atm2.q[ig][atm2.np] = atm.q[ig][ip];
00159                 for (int iw = 0; iw < ctl->nw; iw++)
00160                     atm2.k[iw][atm2.np] = atm.k[iw][ip];
00161                 atm2.np++;
00162             }
00163
00164         /* Get observation data... */
00165         obs2.nr = 1;
00166         obs2.time[0] = obs.time[ir];
00167         obs2.vpz[0] = obs.vpz[ir];
00168         obs2.vplon[0] = obs.vplon[ir];
00169         obs2.vplat[0] = obs.vplat[ir];
00170         obs2.obsz[0] = obs.obsz[ir];
00171         obs2.obslon[0] = obs.obslon[ir];
00172         obs2.obslat[0] = obs.obslat[ir];
00173
00174         /* Check number of data points... */
00175         if (atm2.np > 0) {
00176
00177             /* Call forward model... */
00178             formod(ctl, &atm2, &obs2);
00179
00180             /* Save radiance data... */
00181             for (int id = 0; id < ctl->nd; id++) {
00182                 obs.rad[id][ir] = obs2.rad[id][0];
00183                 obs.tau[id][ir] = obs2.tau[id][0];
00184             }
00185         }
00186     }
00187
00188     /* Write radiance data... */
00189     write_obs(wrkdir, radfile, ctl, &obs);
00190 }
00191
00192 /* Compute single profile... */
00193 else {
00194
00195     /* Call forward model... */
00196     formod(ctl, &atm, &obs);
00197
00198     /* Save radiance data... */
00199     write_obs(wrkdir, radfile, ctl, &obs);
00200
00201     /* Compute contributions... */
00202     if (task[0] == 'c' || task[0] == 'C') {
00203
00204         /* Switch off continua... */
00205         ctl->ctm_co2 = 0;
00206         ctl->ctm_h2o = 0;
00207         ctl->ctm_n2 = 0;
00208         ctl->ctm_o2 = 0;
00209
00210         /* Loop over emitters... */
00211         for (int ig = 0; ig < ctl->ng; ig++) {
00212
00213             /* Copy atmospheric data... */
00214             copy_atm(ctl, &atm2, &atm, 0);
00215
00216             /* Set extinction to zero... */
00217             for (int iw = 0; iw < ctl->nw; iw++)
00218                 for (int ip = 0; ip < atm2.np; ip++)

```

```

00219         atm2.k[iw][ip] = 0;
00220
00221         /* Set volume mixing ratios to zero... */
00222         for (int ig2 = 0; ig2 < ctl->ng; ig2++)
00223             if (ig2 != ig)
00224                 for (int ip = 0; ip < atm2.np; ip++)
00225                     atm2.q[ig2][ip] = 0;
00226
00227         /* Call forward model... */
00228         formod(ctl, &atm2, &obs);
00229
00230         /* Save radiance data... */
00231         sprintf(filename, "%s.%s", radfile, ctl->emitter[ig]);
00232         write_obs(wrkdir, filename, ctl, &obs);
00233     }
00234
00235     /* Copy atmospheric data... */
00236     copy_atm(ctl, &atm2, &atm, 0);
00237
00238     /* Set volume mixing ratios to zero... */
00239     for (int ig = 0; ig < ctl->ng; ig++)
00240         for (int ip = 0; ip < atm2.np; ip++)
00241             atm2.q[ig][ip] = 0;
00242
00243     /* Call forward model... */
00244     formod(ctl, &atm2, &obs);
00245
00246     /* Save radiance data... */
00247     sprintf(filename, "%s.EXTINCT", radfile);
00248     write_obs(wrkdir, filename, ctl, &obs);
00249 }
00250
00251 /* Measure CPU-time... */
00252 if (task[0] == 't' || task[0] == 'T') {
00253
00254     /* Init... */
00255     double t_min, t_max, t_mean = 0, t_sigma = 0;
00256     int n = 0;
00257
00258     /* Initialize random number generator... */
00259     gsl_rng_env_setup();
00260     gsl_rng *rng = gsl_rng_alloc(gsl_rng_default);
00261
00262     /* Loop over profiles... */
00263     do {
00264
00265         /* Create random atmosphere... */
00266         copy_atm(ctl, &atm2, &atm, 0);
00267         double dtemp = 40. * (gsl_rng_uniform(rng) - 0.5);
00268         double dpress = 1. - 0.1 * gsl_rng_uniform(rng);
00269         double dq[NG];
00270         for (int ig = 0; ig < ctl->ng; ig++)
00271             dq[ig] = 0.8 + 0.4 * gsl_rng_uniform(rng);
00272         for (int ip = 0; ip < atm2.np; ip++) {
00273             atm.t[ip] += dtemp;
00274             atm.p[ip] *= dpress;
00275             for (int ig = 0; ig < ctl->ng; ig++)
00276                 atm.q[ig][ip] *= dq[ig];
00277         }
00278
00279         /* Measure runtime... */
00280         double t0 = omp_get_wtime();
00281         formod(ctl, &atm2, &obs);
00282         double dt = omp_get_wtime() - t0;
00283
00284         /* Get runtime statistics... */
00285         t_mean += (dt);
00286         t_sigma += POW2(dt);
00287         if (n == 0 || dt < t_min)
00288             t_min = dt;
00289         if (n == 0 || dt > t_max)
00290             t_max = dt;
00291         n++;
00292     } while (t_mean < 10.0);
00293
00294     /* Write results... */
00295     t_mean /= (double) n;
00296     t_sigma = sqrt(t_sigma / (double) n - POW2(t_mean));
00297     printf("RUNTIME_MEAN = %g s\n", t_mean);
00298     printf("RUNTIME_SIGMA = %g s\n", t_sigma);
00299     printf("RUNTIME_MIN = %g s\n", t_min);
00300     printf("RUNTIME_MAX = %g s\n", t_max);
00301     printf("RAYS_PER_SECOND = %g", (double) obs.nr / t_mean);
00302 }
00303
00304
00305 /* Analyze effect of step size... */

```

```

00306     if (task[0] == 's' || task[0] == 'S') {
00307
00308         /* Reference run... */
00309         ctl->rayds = 0.1;
00310         ctl->raydz = 0.01;
00311         formod(ctl, &atm, &obs);
00312         copy_obs(ctl, &obs2, &obs, 0);
00313
00314         /* Loop over step size... */
00315         for (double dz = 0.01; dz <= 2; dz *= 1.1) {
00316             printf("STEPsize: \n");
00317             for (double ds = 0.1; ds <= 50; ds *= 1.1) {
00318
00319                 /* Set step size... */
00320                 ctl->rayds = ds;
00321                 ctl->raydz = dz;
00322
00323                 /* Measure runtime... */
00324                 double t0 = omp_get_wtime();
00325                 formod(ctl, &atm, &obs);
00326                 double dt = omp_get_wtime() - t0;
00327
00328                 /* Get differences... */
00329                 double mean[ND], sigma[ND];
00330                 for (int id = 0; id < ctl->nd; id++) {
00331                     mean[id] = sigma[id] = 0;
00332                     int n = 0;
00333                     for (int ir = 0; ir < obs.nr; ir++) {
00334                         double err = 200. * (obs.rad[id][ir] - obs2.rad[id][ir])
00335                             / (obs.rad[id][ir] + obs2.rad[id][ir]);
00336                         mean[id] += err;
00337                         sigma[id] += POW2(err);
00338                         n++;
00339                     }
00340                     mean[id] /= n;
00341                     sigma[id] = sqrt(sigma[id] / n - POW2(mean[id]));
00342                 }
00343
00344                 /* Write results... */
00345                 printf("STEPsize: %g %g %g", ds, dz, dt);
00346                 for (int id = 0; id < ctl->nd; id++)
00347                     printf(" %g %g", mean[id], sigma[id]);
00348                 printf("\n");
00349             }
00350         }
00351     }
00352 }
00353 }

```

## 5.9 hydrostatic.c File Reference

Recalculate pressure based on hydrostatic equilibrium.

```
#include "jurassic.h"
```

### Functions

- int [main](#) (int argc, char \*argv[])

#### 5.9.1 Detailed Description

Recalculate pressure based on hydrostatic equilibrium.

Definition in file [hydrostatic.c](#).

#### 5.9.2 Function Documentation

```

5.9.2.1 main() int main (
                int argc,
                char * argv[] )

```

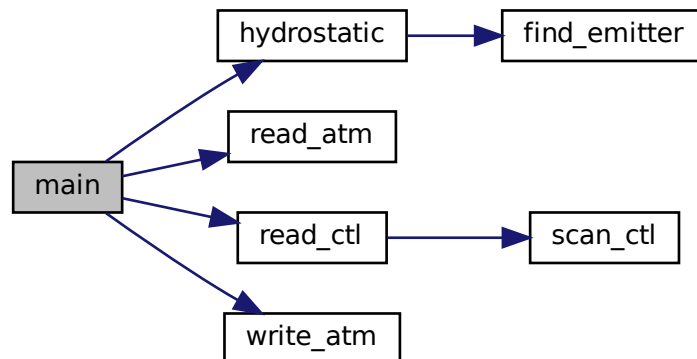
Definition at line 27 of file [hydrostatic.c](#).

```

00029     {
00030
00031     static atm_t atm;
00032     static ctl_t ctl;
00033
00034     /* Check arguments... */
00035     if (argc < 4)
00036         ERRMSG("Give parameters: <ctl> <atm_in> <atm_hyd>");
00037
00038     /* Read control parameters... */
00039     read_ctl(argc, argv, &ctl);
00040
00041     /* Check reference height... */
00042     if (ctl.hydz < 0)
00043         ERRMSG("Set HYDZ>=0!");
00044
00045     /* Read atmospheric data... */
00046     read_atm(NULL, argv[2], &ctl, &atm);
00047
00048     /* Build atmosphere based on hydrostatic equilibrium... */
00049     hydrostatic(&ctl, &atm);
00050
00051     /* Write atmospheric data... */
00052     write_atm(NULL, argv[3], &ctl, &atm);
00053
00054     return EXIT_SUCCESS;
00055 }

```

Here is the call graph for this function:



## 5.10 hydrostatic.c

```

00001  /*
00002   This file is part of JURASSIC.
00003
00004   JURASSIC is free software: you can redistribute it and/or modify
00005   it under the terms of the GNU General Public License as published by
00006   the Free Software Foundation, either version 3 of the License, or
00007   (at your option) any later version.
00008
00009   JURASSIC is distributed in the hope that it will be useful,
00010   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012   GNU General Public License for more details.
00013
00014   You should have received a copy of the GNU General Public License

```

```
00015    along with JURASSIC. If not, see <http://www.gnu.org/licenses/>.
00016
00017    Copyright (C) 2003-2021 Forschungszentrum Juelich GmbH
00018 */
00019
00025 #include "jurassic.h"
00026
00027 int main(
00028     int argc,
00029     char *argv[]) {
00030
00031     static atm_t atm;
00032     static ctl_t ctl;
00033
00034     /* Check arguments... */
00035     if (argc < 4)
00036         ERRMSG("Give parameters: <ctl> <atm_in> <atm_hyd>");
00037
00038     /* Read control parameters... */
00039     read_ctl(argc, argv, &ctl);
00040
00041     /* Check reference height... */
00042     if (ctl.hydz < 0)
00043         ERRMSG("Set HYDZ>=0!");
00044
00045     /* Read atmospheric data... */
00046     read_atm(NULL, argv[2], &ctl, &atm);
00047
00048     /* Build atmosphere based on hydrostatic equilibrium... */
00049     hydrostatic(&ctl, &atm);
00050
00051     /* Write atmospheric data... */
00052     write_atm(NULL, argv[3], &ctl, &atm);
00053
00054     return EXIT_SUCCESS;
00055 }
```

## 5.11 interpolate.c File Reference

Interpolate atmospheric data to another spatial grid.

```
#include "jurassic.h"
```

### Functions

- int [main](#) (int argc, char \*argv[])

#### 5.11.1 Detailed Description

Interpolate atmospheric data to another spatial grid.

Definition in file [interpolate.c](#).

#### 5.11.2 Function Documentation

```

5.11.2.1 main() int main (
                int argc,
                char * argv[] )

```

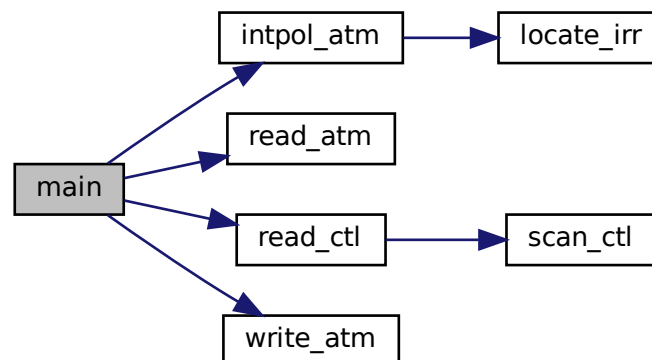
Definition at line 27 of file [interpolate.c](#).

```

00029     {
00030
00031     static atm_t atm_in, atm_pts;
00032     static ctl_t ctl;
00033
00034     double k[NW], q[NG];
00035
00036     /* Interpolate atmospheric data... */
00037
00038     /* Check arguments... */
00039     if (argc < 5)
00040         ERRMSG("Give parameters: <ctl> <atm_in> <atm_pts> <atm_out>");
00041
00042     /* Read control parameters... */
00043     read_ctl(argc, argv, &ctl);
00044
00045     /* Read atmospheric data... */
00046     read_atm(NULL, argv[2], &ctl, &atm_in);
00047     read_atm(NULL, argv[3], &ctl, &atm_pts);
00048
00049     /* Interpolate atmospheric data... */
00050     for (int ip = 0; ip < atm_pts.np; ip++) {
00051         intpol_atm(&ctl, &atm_in, atm_pts.z[ip],
00052                  &atm_pts.p[ip], &atm_pts.t[ip], q, k);
00053         for (int ig = 0; ig < ctl.ng; ig++)
00054             atm_pts.q[ig][ip] = q[ig];
00055         for (int iw = 0; iw < ctl.nw; iw++)
00056             atm_pts.k[iw][ip] = k[iw];
00057     }
00058
00059     /* Save interpolated data... */
00060     write_atm(NULL, argv[4], &ctl, &atm_pts);
00061
00062     return EXIT_SUCCESS;
00063 }

```

Here is the call graph for this function:



## 5.12 interpolate.c

```

00001 /*
00002  This file is part of JURASSIC.
00003
00004  JURASSIC is free software: you can redistribute it and/or modify
00005  it under the terms of the GNU General Public License as published by
00006  the Free Software Foundation, either version 3 of the License, or

```



```

00007      (at your option) any later version.
00008
00009      JURASSIC is distributed in the hope that it will be useful,
00010      but WITHOUT ANY WARRANTY; without even the implied warranty of
00011      MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012      GNU General Public License for more details.
00013
00014      You should have received a copy of the GNU General Public License
00015      along with JURASSIC. If not, see <http://www.gnu.org/licenses/>.
00016
00017      Copyright (C) 2003-2021 Forschungszentrum Juelich GmbH
00018  */
00019
00025  #include "jurassic.h"
00026
00027  int main(
00028      int argc,
00029      char *argv[]) {
00030
00031      static atm_t atm_in, atm_pts;
00032      static ctl_t ctl;
00033
00034      double k[NW], q[NG];
00035
00036      /* Interpolate atmospheric data... */
00037
00038      /* Check arguments... */
00039      if (argc < 5)
00040          ERRMSG("Give parameters: <ctl> <atm_in> <atm_pts> <atm_out>");
00041
00042      /* Read control parameters... */
00043      read_ctl(argc, argv, &ctl);
00044
00045      /* Read atmospheric data... */
00046      read_atm(NULL, argv[2], &ctl, &atm_in);
00047      read_atm(NULL, argv[3], &ctl, &atm_pts);
00048
00049      /* Interpolate atmospheric data... */
00050      for (int ip = 0; ip < atm_pts.np; ip++) {
00051          intpol_atm(&ctl, &atm_in, atm_pts.z[ip],
00052                  &atm_pts.p[ip], &atm_pts.t[ip], q, k);
00053          for (int ig = 0; ig < ctl.ng; ig++)
00054              atm_pts.q[ig][ip] = q[ig];
00055          for (int iw = 0; iw < ctl.nw; iw++)
00056              atm_pts.k[iw][ip] = k[iw];
00057      }
00058
00059      /* Save interpolated data... */
00060      write_atm(NULL, argv[4], &ctl, &atm_pts);
00061
00062      return EXIT_SUCCESS;
00063 }

```

## 5.13 invert.c File Reference

Inversion tool for MPTRAC.

```

#include "jurassic.h"
#include <gsl/gsl_fit.h>

```

### Macros

- #define **NLMAX** 30000000  
*Maximum number of data lines...*
- #define **NMAX** 1000  
*Maximum number of data points...*

### Functions

- int **main** (int argc, char \*argv[])

### 5.13.1 Detailed Description

Inversion tool for MPTRAC.

Definition in file [invert.c](#).

### 5.13.2 Macro Definition Documentation

#### 5.13.2.1 NLMAX `#define NLMAX 30000000`

Maximum number of data lines...

Definition at line 33 of file [invert.c](#).

#### 5.13.2.2 NMAX `#define NMAX 1000`

Maximum number of data points...

Definition at line 36 of file [invert.c](#).

### 5.13.3 Function Documentation

#### 5.13.3.1 `main()` `int main (int argc, char * argv[] )`

Definition at line 42 of file [invert.c](#).

```
00044     {
00045
00046     static ctl_t ctl;
00047
00048     static atm_t atm, atm2;
00049
00050     static obs_t obs;
00051
00052     static gsl_matrix *k;
00053
00054     static FILE *in, *out;
00055
00056     static char line[LEN];
00057
00058     static double rtime[NLMAX], rz[NLMAX], rlon[NLMAX], rlat[NLMAX], obs_meas,
00059     obs_sim, scl = 1.0, scl_old, scl_err, c0, c1, cov00, cov01, cov11,
00060     sumsq, x[NMAX], x2[NMAX], y[NMAX], y_err[NMAX], y2[NMAX], y2_err[NMAX],
00061     y2_sim[NMAX], y2_sim_err[NMAX], w2[NMAX], dt, tol, obs_err;
00062
00063     static float rp[NLMAX], rt[NLMAX], rso2[NLMAX], rh2o[NLMAX],
00064     ro3[NLMAX], robs[NLMAX];
00065
00066     static int data, fit, i, ig, il, ip, it, itmax, n, nl, ndata[NMAX], nprof;
00067
00068     static size_t mk, nk;
00069
00070     /* Check arguments... */
```

```

00071  if (argc < 6)
00072      ERRMSG("Give parameters: <ctl> <prof> <inv> <atm> <kernel>");
00073
00074  /* Read control parameters... */
00075  read_ctl(argc, argv, &ctl);
00076  dt = scan_ctl(argc, argv, "INVERT_DT", -1, "86400", NULL);
00077  obs_err = scan_ctl(argc, argv, "INVERT_OBS_ERR", -1, "1.0", NULL);
00078  data = (int) scan_ctl(argc, argv, "INVERT_DATA", -1, "2", NULL);
00079  fit = (int) scan_ctl(argc, argv, "INVERT_FIT", -1, "3", NULL);
00080  itmax = (int) scan_ctl(argc, argv, "INVERT_ITMAX", -1, "10", NULL);
00081  tol = scan_ctl(argc, argv, "INVERT_TOL", -1, "1e-4", NULL);
00082
00083  /* Check control parameters... */
00084  if (ctl.ng != 4)
00085      ERRMSG("Set NG = 4!");
00086  if (strcmp(ctl.emitter[0], "SO2") != 0)
00087      ERRMSG("Set EMITTER[0] = SO2!");
00088  if (strcmp(ctl.emitter[1], "H2O") != 0)
00089      ERRMSG("Set EMITTER[1] = H2O!");
00090  if (strcmp(ctl.emitter[2], "O3") != 0)
00091      ERRMSG("Set EMITTER[2] = O3!");
00092  if (strcmp(ctl.emitter[3], "CO2") != 0)
00093      ERRMSG("Set EMITTER[3] = CO2!");
00094  if (ctl.nd != 2)
00095      ERRMSG("Set ND = 2!");
00096
00097  /* Set control parameters... */
00098  ctl.write_bbt = 1;
00099  ctl.write_matrix = 1;
00100
00101  /* Set observation data... */
00102  obs.nr = 1;
00103  obs.obsz[0] = 705;
00104
00105  /* -----
00106      Read profiles...
00107      ----- */
00108
00109  /* Read profile data... */
00110  LOG(1, "Read profile data: %s", argv[2]);
00111
00112  /* Open file... */
00113  if (!(in = fopen(argv[2], "r")))
00114      ERRMSG("Cannot open file!");
00115
00116  /* Read file... */
00117  while (fgets(line, LEN, in)) {
00118
00119      /* Read data... */
00120      if (sscanf(line, "%lg %lg %lg %g %g %g %g %g", &rtime[nl],
00121                &rz[nl], &rln[nl], &rlat[nl], &rp[nl], &rt[nl], &rso2[nl],
00122                &rh2o[nl], &ro3[nl], &robs[nl]) != 10)
00123          continue;
00124      if ((++nl) > NLMAX)
00125          ERRMSG("Too many profile data points!");
00126  }
00127
00128  /* Close files... */
00129  fclose(in);
00130
00131  /* -----
00132      Fit scaling factor for total mass...
00133      ----- */
00134
00135  /* Iterations... */
00136  for (it = 0; it < itmax; it++) {
00137
00138      /* Init... */
00139      atm.np = n = 0;
00140      for (i = 0; i < NMAX; i++) {
00141          ndata[i] = 0;
00142          x[i] = y[i] = GSL_NAN;
00143      }
00144
00145      /* Loop over lines... */
00146      for (il = 0; il < nl; il++) {
00147
00148          /* Check for new profile... */
00149          if ((rtime[il] != atm.time[0]
00150              || rlon[il] != atm.lon[0]
00151              || rlat[il] != atm.lat[0])
00152              && atm.np > 0) {
00153
00154              /* Call forward model... */
00155              formod(&ctl, &atm, &obs);
00156              obs_sim = obs.rad[0][0] - obs.rad[1][0];
00157

```

```

00158      /* Get time index... */
00159      i = (int) ((atm.time[0] - rtime[0]) / dt);
00160      if (i < 0 && i >= NMAX)
00161          ERRMSG("Time index out of range!");
00162
00163      /* Get maxima... */
00164      if (data == 1) {
00165          x[i] = (gsl_finite(x[i]) ? GSL_MAX(x[i], obs_sim) : obs_sim);
00166          y[i] = (gsl_finite(y[i]) ? GSL_MAX(y[i], obs_meas) : obs_meas);
00167          y_err[i] = obs_err;
00168          if (gsl_finite(x[i]) && gsl_finite(y[i]))
00169              ndata[i] = 1;
00170      }
00171
00172      /* Get means... */
00173      else if (data == 2) {
00174          if (ndata[i] == 0) {
00175              x[i] = obs_sim;
00176              y[i] = obs_meas;
00177              y_err[i] = POW2(obs_meas);
00178          } else {
00179              x[i] += obs_sim;
00180              y[i] += obs_meas;
00181              y_err[i] += POW2(obs_meas);
00182          }
00183          ndata[i]++;
00184      }
00185
00186      /* Calculate mean atmospheric profile... */
00187      nprof++;
00188      atm2.np = atm.np;
00189      for (ip = 0; ip < atm.np; ip++) {
00190          atm2.time[ip] += atm.time[ip];
00191          atm2.z[ip] += atm.z[ip];
00192          atm2.lon[ip] += atm.lon[ip];
00193          atm2.lat[ip] += atm.lat[ip];
00194          atm2.p[ip] += atm.p[ip];
00195          atm2.t[ip] += atm.t[ip];
00196          for (ig = 1; ig < ctl.ng; ig++)
00197              atm2.q[ig][ip] += atm.q[ig][ip];
00198      }
00199
00200      /* Reset counter... */
00201      atm.np = 0;
00202  }
00203
00204      /* Save data... */
00205      obs_meas = robs[il];
00206      atm.time[atm.np] = rtime[il];
00207      atm.z[atm.np] = rz[il];
00208      atm.lon[atm.np] = rlon[il];
00209      atm.lat[atm.np] = rlat[il];
00210      atm.p[atm.np] = rp[il];
00211      atm.t[atm.np] = rt[il];
00212      atm.q[0][atm.np] = rso2[il] * scl;
00213      atm.q[1][atm.np] = rh2o[il];
00214      atm.q[2][atm.np] = ro3[il];
00215      atm.q[3][atm.np] = 371.789948e-6 + 2.026214e-6
00216          * (atm.time[atm.np] - 63158400.) / 31557600.;
00217      if ((++atm.np) > NP)
00218          ERRMSG("Too many data points!");
00219  }
00220
00221      /* Calculate means... */
00222      if (data == 2)
00223          for (i = 0; i < NMAX; i++)
00224              if (ndata[i] > 0) {
00225                  x[i] /= ndata[i];
00226                  y[i] /= ndata[i];
00227                  y_err[i] = sqrt(GSL_MAX(y_err[i] / ndata[i] - POW2(y[i]), 0.0))
00228                      / sqrt(ndata[i]); /* standard error! */
00229              }
00230
00231      /* Filter data... */
00232      n = 0;
00233      for (i = 0; i < NMAX; i++)
00234          if (ndata[i] > 0 && gsl_finite(x[i]) && gsl_finite(y[i])
00235              && gsl_finite(y_err[i])) {
00236              x2[n] = x[i];
00237              y2[n] = y[i];
00238              y2_err[n] = y_err[i];
00239              w2[n] = 1. / POW2(y_err[i]);
00240              n++;
00241          }
00242
00243      /* Fit radiance data... */
00244      if (fit == 1)

```

```

00245     gsl_fit_mul(x2, 1, y2, 1, (size_t) n, &c1, &cov11, &sumsq);
00246 else if (fit == 2)
00247     gsl_fit_wmul(x2, 1, w2, 1, y2, 1, (size_t) n, &c1, &cov11, &sumsq);
00248 else if (fit == 3)
00249     gsl_fit_linear(x2, 1, y2, 1, (size_t) n, &c0, &c1, &cov00, &cov01,
00250                  &cov11, &sumsq);
00251 else if (fit == 4)
00252     gsl_fit_wlinear(x2, 1, w2, 1, y2, 1, (size_t) n, &c0, &c1, &cov00,
00253                  &cov01, &cov11, &sumsq);
00254 else
00255     ERRMSG("Check INVERT_FIT!");
00256
00257 /* Get new scaling factor... */
00258 scl_old = scl;
00259 scl_err = scl * sqrt(cov11);
00260 scl *= c1;
00261
00262 /* Write info... */
00263 LOG(1, " it= %d | scl= %g +/- %g | RMSE= %g",
00264     it, scl, scl_err, sqrt(sumsq / n));
00265
00266 /* Convergence test... */
00267 if (fabs(2.0 * (scl - scl_old) / (scl + scl_old)) < tol)
00268     break;
00269 }
00270
00271 /* -----
00272 Write inversion data...
00273 ----- */
00274
00275 /* Write info... */
00276 LOG(1, "Write inversion data: %s", argv[3]);
00277
00278 /* Create file... */
00279 if (!(out = fopen(argv[3], "w")))
00280     ERRMSG("Cannot create file!");
00281
00282 /* Write header... */
00283 fprintf(out,
00284     "# $1 = time (seconds since 2000-01-01T00:00Z)\n"
00285     "# $2 = simulated SO2 index [K]\n"
00286     "# $3 = scaled simulated SO2 index [K]\n"
00287     "# $4 = error of scaled simulated SO2 index [K]\n"
00288     "# $5 = observed SO2 index [K]\n"
00289     "# $6 = error of observed SO2 index [K]\n\n");
00290
00291 /* Write data... */
00292 for (i = 0; i < n; i++) {
00293
00294     /* Calculate scaled SO2 index... */
00295     if (fit == 1 || fit == 2)
00296         gsl_fit_mul_est(x2[i], c1, cov11, &y2_sim[i], &y2_sim_err[i]);
00297     else if (fit == 3 || fit == 4)
00298         gsl_fit_linear_est(x2[i], c0, c1, cov00, cov01, cov11, &y2_sim[i],
00299                          &y2_sim_err[i]);
00300
00301     /* Write output... */
00302     fprintf(out, "%.2f %g %g %g %g\n", rtime[0] + (i + 0.5) * dt,
00303         x2[i], y2_sim[i], y2_sim_err[i], y2[i], y2_err[i]);
00304 }
00305
00306 /* Report scaling factor for total mass... */
00307 fprintf(out, "\n");
00308 fprintf(out, "# scl= %g +/- %g\n", scl, scl_err);
00309 fprintf(out, "# c1= %g +/- %g\n", c1, sqrt(cov11));
00310 if (fit == 3 || fit == 4) {
00311     fprintf(out, "# c0= %g +/- %g\n", c0, sqrt(cov00));
00312     fprintf(out, "# corr= %g\n", cov01 / (sqrt(cov00) * sqrt(cov11)));
00313 }
00314 fprintf(out, "# RMSE= %g\n", sqrt(sumsq / n));
00315 fprintf(out, "# n= %d\n", n);
00316
00317 /* Close files... */
00318 fclose(out);
00319
00320 /* -----
00321 Calculate kernel...
00322 ----- */
00323
00324 /* Set atmospheric data... */
00325 for (ip = 0; ip < atm2.np; ip++) {
00326     atm2.time[ip] /= nprof;
00327     atm2.z[ip] /= nprof;
00328     atm2.lon[ip] /= nprof;
00329     atm2.lat[ip] /= nprof;
00330     atm2.p[ip] /= nprof;
00331     atm2.t[ip] /= nprof;

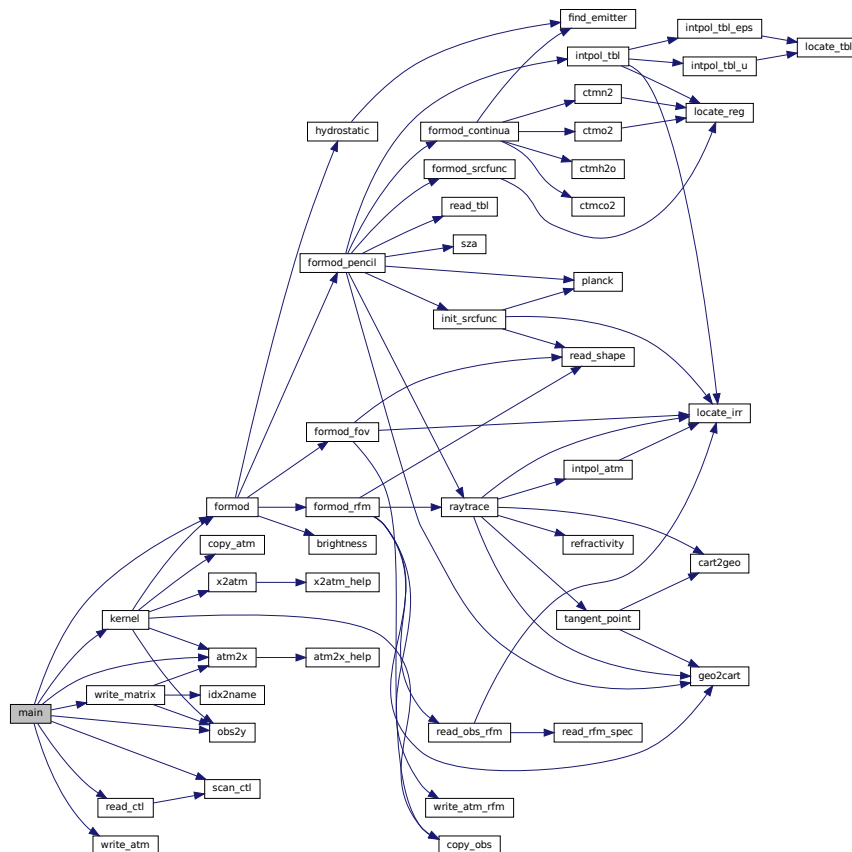
```

```

00332     for (ig = 0; ig < ctl.ng; ig++)
00333     {
00334         atm2.q[ig][ip] /= nprof;
00335     }
00336     /* Get sizes... */
00337     nk = atm2x(&ctl, &atm2, NULL, NULL, NULL);
00338     mk = obs2y(&ctl, &obs, NULL, NULL, NULL);
00339
00340     /* Allocate... */
00341     k = gsl_matrix_alloc(mk, nk);
00342
00343     /* Compute kernel matrix... */
00344     kernel(&ctl, &atm2, &obs, k);
00345
00346     /* Write atmospheric data... */
00347     write_atm(NULL, argv[4], &ctl, &atm);
00348
00349     /* Write matrix to file... */
00350     write_matrix(NULL, argv[5], &ctl, k, &atm2, &obs, "y", "x", "r");
00351
00352     /* Free... */
00353     gsl_matrix_free(k);
00354
00355     return EXIT_SUCCESS;
00356 }

```

Here is the call graph for this function:



## 5.14 invert.c

```

00001 /*
00002  This file is part of JURASSIC.
00003
00004  JURASSIC is free software: you can redistribute it and/or modify
00005  it under the terms of the GNU General Public License as published by
00006  the Free Software Foundation, either version 3 of the License, or
00007  (at your option) any later version.

```

```

00008
00009 JURASSIC is distributed in the hope that it will be useful,
00010 but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012 GNU General Public License for more details.
00013
00014 You should have received a copy of the GNU General Public License
00015 along with JURASSIC. If not, see <http://www.gnu.org/licenses/>.
00016
00017 Copyright (C) 2019-2021 Forschungszentrum Juelich GmbH
00018 */
00019
00025 #include "jurassic.h"
00026 #include <gsl/gsl_fit.h>
00027
00028 /* -----
00029 Constants...
00030 ----- */
00031
00033 #define NLMAX 30000000
00034
00036 #define NMAX 1000
00037
00038 /* -----
00039 Main...
00040 ----- */
00041
00042 int main(
00043     int argc,
00044     char *argv[]) {
00045
00046     static ctl_t ctl;
00047
00048     static atm_t atm, atm2;
00049
00050     static obs_t obs;
00051
00052     static gsl_matrix *k;
00053
00054     static FILE *in, *out;
00055
00056     static char line[LEN];
00057
00058     static double rtime[NLMAX], rz[NLMAX], rlon[NLMAX], rlat[NLMAX], obs_meas,
00059         obs_sim, scl = 1.0, scl_old, scl_err, c0, c1, cov00, cov01, cov11,
00060         sumsq, x[NMAX], x2[NMAX], y[NMAX], y_err[NMAX], y2[NMAX], y2_err[NMAX],
00061         y2_sim[NMAX], y2_sim_err[NMAX], w2[NMAX], dt, tol, obs_err;
00062
00063     static float rp[NLMAX], rt[NLMAX], rso2[NLMAX], rh2o[NLMAX],
00064         ro3[NLMAX], robs[NLMAX];
00065
00066     static int data, fit, i, ig, il, ip, it, itmax, n, nl, ndata[NMAX], nprof;
00067
00068     static size_t mk, nk;
00069
00070     /* Check arguments... */
00071     if (argc < 6)
00072         ERRMSG("Give parameters: <ctl> <prof> <inv> <atm> <kernel>");
00073
00074     /* Read control parameters... */
00075     read_ctl(argc, argv, &ctl);
00076     dt = scan_ctl(argc, argv, "INVERT_DT", -1, "86400", NULL);
00077     obs_err = scan_ctl(argc, argv, "INVERT_OBS_ERR", -1, "1.0", NULL);
00078     data = (int) scan_ctl(argc, argv, "INVERT_DATA", -1, "2", NULL);
00079     fit = (int) scan_ctl(argc, argv, "INVERT_FIT", -1, "3", NULL);
00080     itmax = (int) scan_ctl(argc, argv, "INVERT_ITMAX", -1, "10", NULL);
00081     tol = scan_ctl(argc, argv, "INVERT_TOL", -1, "1e-4", NULL);
00082
00083     /* Check control parameters... */
00084     if (ctl.ng != 4)
00085         ERRMSG("Set NG = 4!");
00086     if (strcmp(ctl.emitter[0], "SO2") != 0)
00087         ERRMSG("Set EMITTER[0] = SO2!");
00088     if (strcmp(ctl.emitter[1], "H2O") != 0)
00089         ERRMSG("Set EMITTER[1] = H2O!");
00090     if (strcmp(ctl.emitter[2], "O3") != 0)
00091         ERRMSG("Set EMITTER[2] = O3!");
00092     if (strcmp(ctl.emitter[3], "CO2") != 0)
00093         ERRMSG("Set EMITTER[3] = CO2!");
00094     if (ctl.nd != 2)
00095         ERRMSG("Set ND = 2!");
00096
00097     /* Set control parameters... */
00098     ctl.write_bbt = 1;
00099     ctl.write_matrix = 1;
00100
00101     /* Set observation data... */

```

```

00102     obs.nr = 1;
00103     obs.obsz[0] = 705;
00104
00105     /* -----
00106     Read profiles...
00107     ----- */
00108
00109     /* Read profile data... */
00110     LOG(1, "Read profile data: %s", argv[2]);
00111
00112     /* Open file... */
00113     if (!(in = fopen(argv[2], "r")))
00114         ERRMSG("Cannot open file!");
00115
00116     /* Read file... */
00117     while (fgets(line, LEN, in)) {
00118
00119         /* Read data... */
00120         if (sscanf(line, "%lg %lg %lg %lg %g %g %g %g %g", &rtime[nl],
00121                     &rz[nl], &rln[nl], &rlat[nl], &rp[nl], &rt[nl], &rso2[nl],
00122                     &rh2o[nl], &ro3[nl], &robs[nl]) != 10)
00123             continue;
00124         if ((++nl) > NLMAX)
00125             ERRMSG("Too many profile data points!");
00126     }
00127
00128     /* Close files... */
00129     fclose(in);
00130
00131     /* -----
00132     Fit scaling factor for total mass...
00133     ----- */
00134
00135     /* Iterations... */
00136     for (it = 0; it < itmax; it++) {
00137
00138         /* Init... */
00139         atm.np = n = 0;
00140         for (i = 0; i < NMAX; i++) {
00141             ndata[i] = 0;
00142             x[i] = y[i] = GSL_NAN;
00143         }
00144
00145         /* Loop over lines... */
00146         for (il = 0; il < nl; il++) {
00147
00148             /* Check for new profile... */
00149             if ((rtime[il] != atm.time[0]
00150                 || rlon[il] != atm.lon[0]
00151                 || rlat[il] != atm.lat[0])
00152                 && atm.np > 0) {
00153
00154                 /* Call forward model... */
00155                 formod(&ctl, &atm, &obs);
00156                 obs_sim = obs.rad[0][0] - obs.rad[1][0];
00157
00158                 /* Get time index... */
00159                 i = (int) ((atm.time[0] - rtime[0]) / dt);
00160                 if (i < 0 && i >= NMAX)
00161                     ERRMSG("Time index out of range!");
00162
00163                 /* Get maxima... */
00164                 if (data == 1) {
00165                     x[i] = (gsl_finite(x[i]) ? GSL_MAX(x[i], obs_sim) : obs_sim);
00166                     y[i] = (gsl_finite(y[i]) ? GSL_MAX(y[i], obs_meas) : obs_meas);
00167                     y_err[i] = obs_err;
00168                     if (gsl_finite(x[i]) && gsl_finite(y[i]))
00169                         ndata[i] = 1;
00170                 }
00171
00172                 /* Get means... */
00173                 else if (data == 2) {
00174                     if (ndata[i] == 0) {
00175                         x[i] = obs_sim;
00176                         y[i] = obs_meas;
00177                         y_err[i] = POW2(obs_meas);
00178                     } else {
00179                         x[i] += obs_sim;
00180                         y[i] += obs_meas;
00181                         y_err[i] += POW2(obs_meas);
00182                     }
00183                     ndata[i]++;
00184                 }
00185
00186                 /* Calculate mean atmospheric profile... */
00187                 nprof++;
00188                 atm2.np = atm.np;

```



```

00189     for (ip = 0; ip < atm.np; ip++) {
00190         atm2.time[ip] += atm.time[ip];
00191         atm2.z[ip] += atm.z[ip];
00192         atm2.lon[ip] += atm.lon[ip];
00193         atm2.lat[ip] += atm.lat[ip];
00194         atm2.p[ip] += atm.p[ip];
00195         atm2.t[ip] += atm.t[ip];
00196         for (ig = 1; ig < ctl.ng; ig++)
00197             atm2.q[ig][ip] += atm.q[ig][ip];
00198     }
00199
00200     /* Reset counter... */
00201     atm.np = 0;
00202 }
00203
00204 /* Save data... */
00205 obs_meas = robs[il];
00206 atm.time[atm.np] = rtime[il];
00207 atm.z[atm.np] = rz[il];
00208 atm.lon[atm.np] = rlon[il];
00209 atm.lat[atm.np] = rlat[il];
00210 atm.p[atm.np] = rp[il];
00211 atm.t[atm.np] = rt[il];
00212 atm.q[0][atm.np] = rso2[il] * scl;
00213 atm.q[1][atm.np] = rh2o[il];
00214 atm.q[2][atm.np] = ro3[il];
00215 atm.q[3][atm.np] = 371.789948e-6 + 2.026214e-6
00216     * (atm.time[atm.np] - 63158400.) / 31557600.;
00217 if (++atm.np > NP)
00218     ERRMSG("Too many data points!");
00219 }
00220
00221 /* Calculate means... */
00222 if (data == 2)
00223     for (i = 0; i < NMAX; i++)
00224         if (ndata[i] > 0) {
00225             x[i] /= ndata[i];
00226             y[i] /= ndata[i];
00227             y_err[i] = sqrt(GSL_MAX(y_err[i] / ndata[i] - POW2(y[i]), 0.0))
00228                 / sqrt(ndata[i]); /* standard error! */
00229         }
00230
00231 /* Filter data... */
00232 n = 0;
00233 for (i = 0; i < NMAX; i++)
00234     if (ndata[i] > 0 && gsl_finite(x[i]) && gsl_finite(y[i])
00235         && gsl_finite(y_err[i])) {
00236         x2[n] = x[i];
00237         y2[n] = y[i];
00238         y2_err[n] = y_err[i];
00239         w2[n] = 1. / POW2(y_err[i]);
00240         n++;
00241     }
00242
00243 /* Fit radiance data... */
00244 if (fit == 1)
00245     gsl_fit_mul(x2, 1, y2, 1, (size_t) n, &c1, &cov11, &sumsq);
00246 else if (fit == 2)
00247     gsl_fit_wmul(x2, 1, w2, 1, y2, 1, (size_t) n, &c1, &cov11, &sumsq);
00248 else if (fit == 3)
00249     gsl_fit_linear(x2, 1, y2, 1, (size_t) n, &c0, &c1, &cov00, &cov01,
00250         &cov11, &sumsq);
00251 else if (fit == 4)
00252     gsl_fit_wlinear(x2, 1, w2, 1, y2, 1, (size_t) n, &c0, &c1, &cov00,
00253         &cov01, &cov11, &sumsq);
00254 else
00255     ERRMSG("Check INVERT_FIT!");
00256
00257 /* Get new scaling factor... */
00258 scl_old = scl;
00259 scl_err = scl * sqrt(cov11);
00260 scl *= c1;
00261
00262 /* Write info... */
00263 LOG(1, " it= %d | scl= %g +/- %g | RMSE= %g",
00264     it, scl, scl_err, sqrt(sumsq / n));
00265
00266 /* Convergence test... */
00267 if (fabs(2.0 * (scl - scl_old) / (scl + scl_old)) < tol)
00268     break;
00269 }
00270
00271 /* -----
00272     Write inversion data...
00273     ----- */
00274
00275 /* Write info... */

```

```

00276 LOG(1, "Write inversion data: %s", argv[3]);
00277
00278 /* Create file... */
00279 if (!out = fopen(argv[3], "w"))
00280     ERRMSG("Cannot create file!");
00281
00282 /* Write header... */
00283 fprintf(out,
00284         "# $1 = time (seconds since 2000-01-01T00:00Z)\n"
00285         "# $2 = simulated SO2 index [K]\n"
00286         "# $3 = scaled simulated SO2 index [K]\n"
00287         "# $4 = error of scaled simulated SO2 index [K]\n"
00288         "# $5 = observed SO2 index [K]\n"
00289         "# $6 = error of observed SO2 index [K]\n\n");
00290
00291 /* Write data... */
00292 for (i = 0; i < n; i++) {
00293
00294     /* Calculate scaled SO2 index... */
00295     if (fit == 1 || fit == 2)
00296         gsl_fit_mul_est(x2[i], c1, cov11, &y2_sim[i], &y2_sim_err[i]);
00297     else if (fit == 3 || fit == 4)
00298         gsl_fit_linear_est(x2[i], c0, c1, cov00, cov01, cov11, &y2_sim[i],
00299                             &y2_sim_err[i]);
00300
00301     /* Write output... */
00302     fprintf(out, "%.2f %g %g %g %g %g\n", rtime[0] + (i + 0.5) * dt,
00303             x2[i], y2_sim[i], y2_sim_err[i], y2[i], y2_err[i]);
00304 }
00305
00306 /* Report scaling factor for total mass... */
00307 fprintf(out, "\n");
00308 fprintf(out, "#      scl= %g +/- %g\n", scl, scl_err);
00309 fprintf(out, "#      c1= %g +/- %g\n", c1, sqrt(cov11));
00310 if (fit == 3 || fit == 4) {
00311     fprintf(out, "#      c0= %g +/- %g\n", c0, sqrt(cov00));
00312     fprintf(out, "#      corr= %g\n", cov01 / (sqrt(cov00) * sqrt(cov11)));
00313 }
00314 fprintf(out, "#      RMSE= %g\n", sqrt(sumsq / n));
00315 fprintf(out, "#      n= %d\n", n);
00316
00317 /* Close files... */
00318 fclose(out);
00319
00320 /* -----
00321 Calculate kernel...
00322 ----- */
00323
00324 /* Set atmospheric data... */
00325 for (ip = 0; ip < atm2.np; ip++) {
00326     atm2.time[ip] /= nprof;
00327     atm2.z[ip] /= nprof;
00328     atm2.lon[ip] /= nprof;
00329     atm2.lat[ip] /= nprof;
00330     atm2.p[ip] /= nprof;
00331     atm2.t[ip] /= nprof;
00332     for (ig = 0; ig < ctl.ng; ig++)
00333         atm2.q[ig][ip] /= nprof;
00334 }
00335
00336 /* Get sizes... */
00337 nk = atm2x(&ctl, &atm2, NULL, NULL, NULL);
00338 mk = obs2y(&ctl, &obs, NULL, NULL, NULL);
00339
00340 /* Allocate... */
00341 k = gsl_matrix_alloc(mk, nk);
00342
00343 /* Compute kernel matrix... */
00344 kernel(&ctl, &atm2, &obs, k);
00345
00346 /* Write atmospheric data... */
00347 write_atm(NULL, argv[4], &ctl, &atm);
00348
00349 /* Write matrix to file... */
00350 write_matrix(NULL, argv[5], &ctl, k, &atm2, &obs, "y", "x", "r");
00351
00352 /* Free... */
00353 gsl_matrix_free(k);
00354
00355 return EXIT_SUCCESS;
00356 }

```

## 5.15 jsec2time.c File Reference

Convert Julian seconds to date.

```
#include "jurassic.h"
```

### Functions

- int [main](#) (int argc, char \*argv[ ])

#### 5.15.1 Detailed Description

Convert Julian seconds to date.

Definition in file [jsec2time.c](#).

#### 5.15.2 Function Documentation

**5.15.2.1 main()** int main (  
    int argc,  
    char \* argv[ ] )

Definition at line 27 of file [jsec2time.c](#).

```
00029     {  
00030  
00031     double remain;  
00032  
00033     int day, hour, min, mon, sec, year;  
00034  
00035     /* Check arguments... */  
00036     if (argc < 2)  
00037         ERRMSG("Give parameters: <jsec>");  
00038  
00039     /* Read arguments... */  
00040     double jsec = atof(argv[1]);  
00041  
00042     /* Convert time... */  
00043     jsec2time(jsec, &year, &mon, &day, &hour, &min, &sec, &remain);  
00044     printf("%d %d %d %d %d %d %g\n", year, mon, day, hour, min, sec, remain);  
00045  
00046     return EXIT_SUCCESS;  
00047 }
```

Here is the call graph for this function:



## 5.16 jsec2time.c

```

00001 /*
00002  This file is part of JURASSIC.
00003
00004  JURASSIC is free software: you can redistribute it and/or modify
00005  it under the terms of the GNU General Public License as published by
00006  the Free Software Foundation, either version 3 of the License, or
00007  (at your option) any later version.
00008
00009  JURASSIC is distributed in the hope that it will be useful,
00010  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  GNU General Public License for more details.
00013
00014  You should have received a copy of the GNU General Public License
00015  along with JURASSIC. If not, see <http://www.gnu.org/licenses/>.
00016
00017  Copyright (C) 2003-2021 Forschungszentrum Juelich GmbH
00018 */
00019
00025 #include "jurassic.h"
00026
00027 int main(
00028     int argc,
00029     char *argv[]) {
00030
00031     double remain;
00032
00033     int day, hour, min, mon, sec, year;
00034
00035     /* Check arguments... */
00036     if (argc < 2)
00037         ERRMSG("Give parameters: <jsec>");
00038
00039     /* Read arguments... */
00040     double jsec = atof(argv[1]);
00041
00042     /* Convert time... */
00043     jsec2time(jsec, &year, &mon, &day, &hour, &min, &sec, &remain);
00044     printf("%d %d %d %d %d %d %g\n", year, mon, day, hour, min, sec, remain);
00045
00046     return EXIT_SUCCESS;
00047 }

```

## 5.17 jurassic.c File Reference

JURASSIC library definitions.

```
#include "jurassic.h"
```

### Functions

- `size_t atm2x (ctl_t *ctl, atm_t *atm, gsl_vector *x, int *iqa, int *ipa)`  
*Compose state vector or parameter vector.*
- `void atm2x_help (double value, int value_iqa, int value_ip, gsl_vector *x, int *iqa, int *ipa, size_t *n)`  
*Add element to state vector.*
- `double brightness (double rad, double nu)`  
*Compute brightness temperature.*
- `void cart2geo (double *x, double *z, double *lon, double *lat)`  
*Convert Cartesian coordinates to geolocation.*
- `void climatology (ctl_t *ctl, atm_t *atm)`  
*Interpolate climatological data.*
- `double ctmcO2 (double nu, double p, double t, double u)`  
*Compute carbon dioxide continuum (optical depth).*
- `double ctmh2O (double nu, double p, double t, double q, double u)`  
*Compute water vapor continuum (optical depth).*

- double `ctmn2` (double nu, double p, double t)  
*Compute nitrogen continuum (absorption coefficient).*
- double `ctmo2` (double nu, double p, double t)  
*Compute oxygen continuum (absorption coefficient).*
- void `copy_atm` (ctl\_t \*ctl, atm\_t \*atm\_dest, atm\_t \*atm\_src, int init)  
*Copy and initialize atmospheric data.*
- void `copy_obs` (ctl\_t \*ctl, obs\_t \*obs\_dest, obs\_t \*obs\_src, int init)  
*Copy and initialize observation data.*
- int `find_emitter` (ctl\_t \*ctl, const char \*emitter)  
*Find index of an emitter.*
- void `formod` (ctl\_t \*ctl, atm\_t \*atm, obs\_t \*obs)  
*Determine ray paths and compute radiative transfer.*
- void `formod_continua` (ctl\_t \*ctl, los\_t \*los, int ip, double \*beta)  
*Compute absorption coefficient of continua.*
- void `formod_fov` (ctl\_t \*ctl, obs\_t \*obs)  
*Apply field of view convolution.*
- void `formod_pencil` (ctl\_t \*ctl, atm\_t \*atm, obs\_t \*obs, int ir)  
*Compute radiative transfer for a pencil beam.*
- void `formod_rfm` (ctl\_t \*ctl, atm\_t \*atm, obs\_t \*obs)  
*Apply RFM for radiative transfer calculations.*
- void `formod_srcfunc` (ctl\_t \*ctl, tbl\_t \*tbl, double t, double \*src)  
*Compute Planck source function.*
- void `geo2cart` (double z, double lon, double lat, double \*x)  
*Convert geolocation to Cartesian coordinates.*
- void `hydrostatic` (ctl\_t \*ctl, atm\_t \*atm)  
*Set hydrostatic equilibrium.*
- void `idx2name` (ctl\_t \*ctl, int idx, char \*quantity)  
*Determine name of state vector quantity for given index.*
- void `init_srcfunc` (ctl\_t \*ctl, tbl\_t \*tbl)  
*Initialize source function table.*
- void `intpol_atm` (ctl\_t \*ctl, atm\_t \*atm, double z, double \*p, double \*t, double \*q, double \*k)  
*Interpolate atmospheric data.*
- void `intpol_tbl` (ctl\_t \*ctl, tbl\_t \*tbl, los\_t \*los, int ip, double tau\_path[ND][NG], double tau\_seg[ND])  
*Get transmittance from look-up tables.*
- double `intpol_tbl_eps` (tbl\_t \*tbl, int ig, int id, int ip, int it, double u)  
*Interpolate emissivity from look-up tables.*
- double `intpol_tbl_u` (tbl\_t \*tbl, int ig, int id, int ip, int it, double eps)  
*Interpolate column density from look-up tables.*
- void `jsec2time` (double jsec, int \*year, int \*mon, int \*day, int \*hour, int \*min, int \*sec, double \*remain)  
*Convert seconds to date.*
- void `kernel` (ctl\_t \*ctl, atm\_t \*atm, obs\_t \*obs, gsl\_matrix \*k)  
*Compute Jacobians.*
- int `locate_irr` (double \*xx, int n, double x)  
*Find array index for irregular grid.*
- int `locate_reg` (double \*xx, int n, double x)  
*Find array index for regular grid.*
- int `locate_tbl` (float \*xx, int n, double x)  
*Find array index in float array.*
- size\_t `obs2y` (ctl\_t \*ctl, obs\_t \*obs, gsl\_vector \*y, int \*ida, int \*ira)  
*Compose measurement vector.*
- double `planck` (double t, double nu)

- Compute Planck function.*

  - void `raytrace` (`ctl_t` \*ctl, `atm_t` \*atm, `obs_t` \*obs, `los_t` \*los, int ir)
- Do ray-tracing to determine LOS.*

  - void `read_atm` (const char \*dirname, const char \*filename, `ctl_t` \*ctl, `atm_t` \*atm)
- Read atmospheric data.*

  - void `read_ctl` (int argc, char \*argv[], `ctl_t` \*ctl)
- Read forward model control parameters.*

  - void `read_matrix` (const char \*dirname, const char \*filename, gsl\_matrix \*matrix)
- Read matrix.*

  - void `read_obs` (const char \*dirname, const char \*filename, `ctl_t` \*ctl, `obs_t` \*obs)
- Read observation data.*

  - double `read_obs_rfm` (const char \*basename, double z, double \*nu, double \*f, int n)
- Read observation data in RFM format.*

  - void `read_rfm_spec` (const char \*filename, double \*nu, double \*rad, int \*npts)
- Read RFM spectrum.*

  - void `read_shape` (const char \*filename, double \*x, double \*y, int \*n)
- Read shape function.*

  - void `read_tbl` (`ctl_t` \*ctl, `tbl_t` \*tbl)
- Read look-up table data.*

  - double `refractivity` (double p, double t)
- Compute refractivity (return value is  $n - 1$ ).*

  - double `scan_ctl` (int argc, char \*argv[], const char \*varname, int arridx, const char \*defvalue, char \*value)
- Search control parameter file for variable entry.*

  - double `sza` (double sec, double lon, double lat)
- Calculate solar zenith angle.*

  - void `tangent_point` (`los_t` \*los, double \*tpz, double \*tplon, double \*tplat)
- Find tangent point of a given LOS.*

  - void `time2jsec` (int year, int mon, int day, int hour, int min, int sec, double remain, double \*jsec)
- Convert date to seconds.*

  - void `timer` (const char \*name, const char \*file, const char \*func, int line, int mode)
- Measure wall-clock time.*

  - void `write_atm` (const char \*dirname, const char \*filename, `ctl_t` \*ctl, `atm_t` \*atm)
- Write atmospheric data.*

  - void `write_atm_rfm` (const char \*filename, `ctl_t` \*ctl, `atm_t` \*atm)
- Write atmospheric data in RFM format.*

  - void `write_matrix` (const char \*dirname, const char \*filename, `ctl_t` \*ctl, gsl\_matrix \*matrix, `atm_t` \*atm, `obs_t` \*obs, const char \*rowsep, const char \*colsep, const char \*sort)
- Write matrix.*

  - void `write_obs` (const char \*dirname, const char \*filename, `ctl_t` \*ctl, `obs_t` \*obs)
- Write observation data.*

  - void `write_shape` (const char \*filename, double \*x, double \*y, int n)
- Write shape function.*

  - void `write_tbl` (`ctl_t` \*ctl, `tbl_t` \*tbl)
- Write look-up table data.*

  - void `x2atm` (`ctl_t` \*ctl, gsl\_vector \*x, `atm_t` \*atm)
- Decompose parameter vector or state vector.*

  - void `x2atm_help` (double \*value, gsl\_vector \*x, size\_t \*n)
- Get element from state vector.*

  - void `y2obs` (`ctl_t` \*ctl, gsl\_vector \*y, `obs_t` \*obs)
- Decompose measurement vector.*

### 5.17.1 Detailed Description

JURASSIC library definitions.

Definition in file [jurassic.c](#).

### 5.17.2 Function Documentation

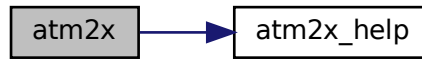
**5.17.2.1 atm2x()** `size_t atm2x (`  
`ctl_t * ctl,`  
`atm_t * atm,`  
`gsl_vector * x,`  
`int * iqa,`  
`int * ipa )`

Compose state vector or parameter vector.

Definition at line 29 of file [jurassic.c](#).

```
00034     {
00035
00036         size_t n = 0;
00037
00038         /* Add pressure... */
00039         for (int ip = 0; ip < atm->np; ip++)
00040             if (atm->z[ip] >= ctl->retp_zmin && atm->z[ip] <= ctl->retp_zmax)
00041                 atm2x_help(atm->p[ip], IDXP, ip, x, iqa, ipa, &n);
00042
00043         /* Add temperature... */
00044         for (int ip = 0; ip < atm->np; ip++)
00045             if (atm->z[ip] >= ctl->rett_zmin && atm->z[ip] <= ctl->rett_zmax)
00046                 atm2x_help(atm->t[ip], IDXT, ip, x, iqa, ipa, &n);
00047
00048         /* Add volume mixing ratios... */
00049         for (int ig = 0; ig < ctl->ng; ig++)
00050             for (int ip = 0; ip < atm->np; ip++)
00051                 if (atm->z[ip] >= ctl->retq_zmin[ig]
00052                     && atm->z[ip] <= ctl->retq_zmax[ig])
00053                     atm2x_help(atm->q[ig][ip], IDXQ(ig), ip, x, iqa, ipa, &n);
00054
00055         /* Add extinction... */
00056         for (int iw = 0; iw < ctl->nw; iw++)
00057             for (int ip = 0; ip < atm->np; ip++)
00058                 if (atm->z[ip] >= ctl->retk_zmin[iw]
00059                     && atm->z[ip] <= ctl->retk_zmax[iw])
00060                     atm2x_help(atm->k[iw][ip], IDXK(iw), ip, x, iqa, ipa, &n);
00061
00062         /* Add cloud parameters... */
00063         if (ctl->ret_clz)
00064             atm2x_help(atm->clz, IDXCLZ, 0, x, iqa, ipa, &n);
00065         if (ctl->ret_cldz)
00066             atm2x_help(atm->cldz, IDXCLDZ, 0, x, iqa, ipa, &n);
00067         if (ctl->ret_clk)
00068             for (int icl = 0; icl < ctl->ncl; icl++)
00069                 atm2x_help(atm->clk[icl], IDXCLK(icl), 0, x, iqa, ipa, &n);
00070
00071         /* Add surface parameters... */
00072         if (ctl->ret_sfz)
00073             atm2x_help(atm->sfz, IDXSFZ, 0, x, iqa, ipa, &n);
00074         if (ctl->ret_sfp)
00075             atm2x_help(atm->sfp, IDXSFP, 0, x, iqa, ipa, &n);
00076         if (ctl->ret_sft)
00077             atm2x_help(atm->sft, IDXSFT, 0, x, iqa, ipa, &n);
00078         if (ctl->ret_sfeps)
00079             for (int isf = 0; isf < ctl->nsf; isf++)
00080                 atm2x_help(atm->sfeps[isf], IDXSFEPS(isf), 0, x, iqa, ipa, &n);
00081         return n;
00082     }
00083 }
```

Here is the call graph for this function:



**5.17.2.2 atm2x\_help()** void atm2x\_help (  
    double value,  
    int value\_iqa,  
    int value\_ip,  
    gsl\_vector \* x,  
    int \* iqa,  
    int \* ipa,  
    size\_t \* n )

Add element to state vector.

Definition at line 87 of file [jurassic.c](#).

```
00094     {  
00095  
00096     /* Add element to state vector... */  
00097     if (x != NULL)  
00098         gsl_vector_set(x, *n, value);  
00099     if (iqa != NULL)  
00100         iqa[*n] = value_iqa;  
00101     if (ipa != NULL)  
00102         ipa[*n] = value_ip;  
00103     (*n)++;  
00104 }
```

**5.17.2.3 brightness()** double brightness (  
    double rad,  
    double nu )

Compute brightness temperature.

Definition at line 109 of file [jurassic.c](#).

```
00111     {  
00112  
00113     return C2 * nu / gsl_log1p(C1 * POW3(nu) / rad);  
00114 }
```



```

5.17.2.4 cart2geo() void cart2geo (
    double * x,
    double * z,
    double * lon,
    double * lat )

```

Convert Cartesian coordinates to geolocation.

Definition at line 119 of file [jurassic.c](#).

```

00123     {
00124
00125     double radius = NORM(x);
00126
00127     *lat = asin(x[2] / radius) * 180 / M_PI;
00128     *lon = atan2(x[1], x[0]) * 180 / M_PI;
00129     *z = radius - RE;
00130 }

```

```

5.17.2.5 climatology() void climatology (
    ctl_t * ctl,
    atm_t * atm )

```

Interpolate climatological data.

Definition at line 134 of file [jurassic.c](#).

```

00136     {
00137
00138     static double z[121] = {
00139         0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
00140         20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37,
00141         38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55,
00142         56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73,
00143         74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91,
00144         92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107,
00145         108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120
00146     };
00147
00148     static double pre[121] = {
00149         1017, 901.083, 796.45, 702.227, 617.614, 541.644, 473.437, 412.288,
00150         357.603, 308.96, 265.994, 228.348, 195.619, 167.351, 143.039, 122.198,
00151         104.369, 89.141, 76.1528, 65.0804, 55.641, 47.591, 40.7233, 34.8637,
00152         29.8633, 25.5956, 21.9534, 18.8445, 16.1909, 13.9258, 11.9913,
00153         10.34, 8.92988, 7.72454, 6.6924, 5.80701, 5.04654, 4.39238, 3.82902,
00154         3.34337, 2.92413, 2.56128, 2.2464, 1.97258, 1.73384, 1.52519, 1.34242,
00155         1.18197, 1.04086, 0.916546, 0.806832, 0.709875, 0.624101, 0.548176,
00156         0.480974, 0.421507, 0.368904, 0.322408, 0.281386, 0.245249, 0.213465,
00157         0.185549, 0.161072, 0.139644, 0.120913, 0.104568, 0.0903249, 0.0779269,
00158         0.0671493, 0.0577962, 0.0496902, 0.0426736, 0.0366093, 0.0313743,
00159         0.0268598, 0.0229699, 0.0196206, 0.0167399, 0.0142646, 0.0121397,
00160         0.0103181, 0.00875775, 0.00742226, 0.00628076, 0.00530519, 0.00447183,
00161         0.00376124, 0.00315632, 0.00264248, 0.00220738, 0.00184003, 0.00153095,
00162         0.00127204, 0.00105608, 0.000876652, 0.00072798, 0.00060492,
00163         0.000503201, 0.000419226, 0.000349896, 0.000292659, 0.000245421,
00164         0.000206394, 0.000174125, 0.000147441, 0.000125333, 0.000106985,
00165         9.173e-05, 7.90172e-05, 6.84172e-05, 5.95574e-05, 5.21183e-05,
00166         4.58348e-05, 4.05127e-05, 3.59987e-05, 3.21583e-05, 2.88718e-05,
00167         2.60322e-05, 2.35687e-05, 2.14263e-05, 1.95489e-05
00168     };
00169
00170     static double tem[121] = {
00171         285.14, 279.34, 273.91, 268.3, 263.24, 256.55, 250.2, 242.82, 236.17,
00172         229.87, 225.04, 221.19, 218.85, 217.19, 216.2, 215.68, 215.42, 215.55,
00173         215.92, 216.4, 216.93, 217.45, 218, 218.68, 219.39, 220.25, 221.3,
00174         222.41, 223.88, 225.42, 227.2, 229.52, 231.89, 234.51, 236.85, 239.42,
00175         241.94, 244.57, 247.36, 250.32, 253.34, 255.82, 258.27, 260.39,
00176         262.03, 263.45, 264.2, 264.78, 264.67, 264.38, 263.24, 262.03, 260.02,
00177         258.09, 255.63, 253.28, 250.43, 247.81, 245.26, 242.77, 240.38,
00178         237.94, 235.79, 233.53, 231.5, 229.53, 227.6, 225.62, 223.77, 222.06,
00179         220.33, 218.69, 217.18, 215.64, 214.13, 212.52, 210.86, 209.25,
00180         207.49, 205.81, 204.11, 202.22, 200.32, 198.39, 195.92, 193.46,
00181         190.94, 188.31, 185.82, 183.57, 181.43, 179.74, 178.64, 178.1, 178.25,
00182         178.7, 179.41, 180.67, 182.31, 184.18, 186.6, 189.53, 192.66, 196.54,
00183         201.13, 205.93, 211.73, 217.86, 225, 233.53, 242.57, 252.14, 261.48,
00184         272.97, 285.26, 299.12, 312.2, 324.17, 338.34, 352.56, 365.28
00185     };

```

```

00186
00187 static double c2h2[121] = {
00188     1.352e-09, 2.83e-10, 1.269e-10, 6.926e-11, 4.346e-11, 2.909e-11,
00189     2.014e-11, 1.363e-11, 8.71e-12, 5.237e-12, 2.718e-12, 1.375e-12,
00190     5.786e-13, 2.16e-13, 7.317e-14, 2.551e-14, 1.055e-14, 4.758e-15,
00191     2.056e-15, 7.703e-16, 2.82e-16, 1.035e-16, 4.382e-17, 1.946e-17,
00192     9.638e-18, 5.2e-18, 2.811e-18, 1.494e-18, 7.925e-19, 4.213e-19,
00193     1.998e-19, 8.78e-20, 3.877e-20, 1.728e-20, 7.743e-21, 3.536e-21,
00194     1.623e-21, 7.508e-22, 3.508e-22, 1.65e-22, 7.837e-23, 3.733e-23,
00195     1.808e-23, 8.77e-24, 4.285e-24, 2.095e-24, 1.032e-24, 5.082e-25,
00196     2.506e-25, 1.236e-25, 6.088e-26, 2.996e-26, 1.465e-26, 0, 0, 0,
00197     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00198     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00199     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
00200 };
00201
00202 static double c2h6[121] = {
00203     2.667e-09, 2.02e-09, 1.658e-09, 1.404e-09, 1.234e-09, 1.109e-09,
00204     1.012e-09, 9.262e-10, 8.472e-10, 7.71e-10, 6.932e-10, 6.216e-10,
00205     5.503e-10, 4.87e-10, 4.342e-10, 3.861e-10, 3.347e-10, 2.772e-10,
00206     2.209e-10, 1.672e-10, 1.197e-10, 8.536e-11, 5.783e-11, 3.846e-11,
00207     2.495e-11, 1.592e-11, 1.017e-11, 6.327e-12, 3.895e-12, 2.403e-12,
00208     1.416e-12, 8.101e-13, 4.649e-13, 2.686e-13, 1.557e-13, 9.14e-14,
00209     5.386e-14, 3.19e-14, 1.903e-14, 1.14e-14, 6.875e-15, 4.154e-15,
00210     2.538e-15, 1.553e-15, 9.548e-16, 5.872e-16, 3.63e-16, 2.244e-16,
00211     1.388e-16, 8.587e-17, 5.308e-17, 3.279e-17, 2.017e-17, 1.238e-17,
00212     7.542e-18, 4.585e-18, 2.776e-18, 1.671e-18, 9.985e-19, 5.937e-19,
00213     3.518e-19, 2.07e-19, 1.215e-19, 7.06e-20, 4.097e-20, 2.37e-20,
00214     1.363e-20, 7.802e-21, 4.441e-21, 2.523e-21, 1.424e-21, 8.015e-22,
00215     4.497e-22, 2.505e-22, 1.391e-22, 7.691e-23, 4.238e-23, 2.331e-23,
00216     1.274e-23, 6.929e-24, 3.752e-24, 2.02e-24, 1.083e-24, 5.774e-25,
00217     3.041e-25, 1.593e-25, 8.308e-26, 4.299e-26, 2.195e-26, 1.112e-26,
00218     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00219     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
00220 };
00221
00222 static double ccl4[121] = {
00223     1.075e-10, 1.075e-10, 1.075e-10, 1.075e-10, 1.075e-10, 1.075e-10,
00224     1.075e-10, 1.075e-10, 1.075e-10, 1.06e-10, 1.024e-10, 9.69e-11,
00225     8.93e-11, 8.078e-11, 7.213e-11, 6.307e-11, 5.383e-11, 4.49e-11,
00226     3.609e-11, 2.705e-11, 1.935e-11, 1.385e-11, 8.35e-12, 5.485e-12,
00227     3.853e-12, 2.22e-12, 5.875e-13, 3.445e-13, 1.015e-13, 6.075e-14,
00228     4.383e-14, 2.692e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00229     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00230     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00231     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00232     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00233     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00234     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00235     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00236     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00237     1e-14, 1e-14, 1e-14
00238 };
00239
00240 static double ch4[121] = {
00241     1.864e-06, 1.835e-06, 1.819e-06, 1.805e-06, 1.796e-06, 1.788e-06,
00242     1.782e-06, 1.776e-06, 1.769e-06, 1.761e-06, 1.749e-06, 1.734e-06,
00243     1.716e-06, 1.692e-06, 1.654e-06, 1.61e-06, 1.567e-06, 1.502e-06,
00244     1.433e-06, 1.371e-06, 1.323e-06, 1.277e-06, 1.232e-06, 1.188e-06,
00245     1.147e-06, 1.108e-06, 1.07e-06, 1.027e-06, 9.854e-07, 9.416e-07,
00246     8.933e-07, 8.478e-07, 7.988e-07, 7.515e-07, 7.07e-07, 6.64e-07,
00247     6.239e-07, 5.864e-07, 5.512e-07, 5.184e-07, 4.87e-07, 4.571e-07,
00248     4.296e-07, 4.04e-07, 3.802e-07, 3.578e-07, 3.383e-07, 3.203e-07,
00249     3.032e-07, 2.889e-07, 2.76e-07, 2.635e-07, 2.519e-07, 2.409e-07,
00250     2.302e-07, 2.219e-07, 2.144e-07, 2.071e-07, 1.999e-07, 1.93e-07,
00251     1.862e-07, 1.795e-07, 1.731e-07, 1.668e-07, 1.607e-07, 1.548e-07,
00252     1.49e-07, 1.434e-07, 1.38e-07, 1.328e-07, 1.277e-07, 1.227e-07,
00253     1.18e-07, 1.134e-07, 1.089e-07, 1.046e-07, 1.004e-07, 9.635e-08,
00254     9.245e-08, 8.867e-08, 8.502e-08, 8.15e-08, 7.809e-08, 7.48e-08,
00255     7.159e-08, 6.849e-08, 6.55e-08, 6.262e-08, 5.98e-08, 5.708e-08,
00256     5.448e-08, 5.194e-08, 4.951e-08, 4.72e-08, 4.5e-08, 4.291e-08,
00257     4.093e-08, 3.905e-08, 3.729e-08, 3.563e-08, 3.408e-08, 3.265e-08,
00258     3.128e-08, 2.996e-08, 2.87e-08, 2.76e-08, 2.657e-08, 2.558e-08,
00259     2.467e-08, 2.385e-08, 2.307e-08, 2.234e-08, 2.168e-08, 2.108e-08,
00260     2.05e-08, 1.998e-08, 1.947e-08, 1.902e-08, 1.86e-08, 1.819e-08,
00261     1.782e-08
00262 };
00263
00264 static double clo[121] = {
00265     7.419e-15, 1.061e-14, 1.518e-14, 2.195e-14, 3.175e-14, 4.666e-14,
00266     6.872e-14, 1.03e-13, 1.553e-13, 2.375e-13, 3.664e-13, 5.684e-13,
00267     8.915e-13, 1.402e-12, 2.269e-12, 4.125e-12, 7.501e-12, 1.257e-11,
00268     2.048e-11, 3.338e-11, 5.44e-11, 8.846e-11, 1.008e-10, 1.082e-10,
00269     1.157e-10, 1.232e-10, 1.312e-10, 1.539e-10, 1.822e-10, 2.118e-10,
00270     2.387e-10, 2.687e-10, 2.875e-10, 3.031e-10, 3.23e-10, 3.648e-10,
00271     4.117e-10, 4.477e-10, 4.633e-10, 4.794e-10, 4.95e-10, 5.104e-10,
00272     5.259e-10, 5.062e-10, 4.742e-10, 4.443e-10, 4.051e-10, 3.659e-10,

```

```
00273 3.305e-10, 2.911e-10, 2.54e-10, 2.215e-10, 1.927e-10, 1.675e-10,
00274 1.452e-10, 1.259e-10, 1.09e-10, 9.416e-11, 8.119e-11, 6.991e-11,
00275 6.015e-11, 5.163e-11, 4.43e-11, 3.789e-11, 3.24e-11, 2.769e-11,
00276 2.361e-11, 2.011e-11, 1.71e-11, 1.453e-11, 1.233e-11, 1.045e-11,
00277 8.851e-12, 7.48e-12, 6.316e-12, 5.326e-12, 4.487e-12, 3.778e-12,
00278 3.176e-12, 2.665e-12, 2.234e-12, 1.87e-12, 1.563e-12, 1.304e-12,
00279 1.085e-12, 9.007e-13, 7.468e-13, 6.179e-13, 5.092e-13, 4.188e-13,
00280 3.442e-13, 2.816e-13, 2.304e-13, 1.885e-13, 1.542e-13, 1.263e-13,
00281 1.035e-13, 8.5e-14, 7.004e-14, 5.783e-14, 4.795e-14, 4.007e-14,
00282 3.345e-14, 2.792e-14, 2.33e-14, 1.978e-14, 1.686e-14, 1.438e-14,
00283 1.234e-14, 1.07e-14, 9.312e-15, 8.131e-15, 7.164e-15, 6.367e-15,
00284 5.67e-15, 5.088e-15, 4.565e-15, 4.138e-15, 3.769e-15, 3.432e-15,
00285 3.148e-15
00286 };
00287
00288 static double clono2[121] = {
00289 1.011e-13, 1.515e-13, 2.272e-13, 3.446e-13, 5.231e-13, 8.085e-13,
00290 1.253e-12, 1.979e-12, 3.149e-12, 5.092e-12, 8.312e-12, 1.366e-11,
00291 2.272e-11, 3.791e-11, 6.209e-11, 9.101e-11, 1.334e-10, 1.951e-10,
00292 2.853e-10, 3.94e-10, 4.771e-10, 5.771e-10, 6.675e-10, 7.665e-10,
00293 8.504e-10, 8.924e-10, 9.363e-10, 8.923e-10, 8.411e-10, 7.646e-10,
00294 6.525e-10, 5.576e-10, 4.398e-10, 3.403e-10, 2.612e-10, 1.915e-10,
00295 1.407e-10, 1.028e-10, 7.455e-11, 5.42e-11, 3.708e-11, 2.438e-11,
00296 1.618e-11, 1.075e-11, 7.17e-12, 4.784e-12, 3.205e-12, 2.147e-12,
00297 1.44e-12, 9.654e-13, 6.469e-13, 4.332e-13, 2.891e-13, 1.926e-13,
00298 1.274e-13, 8.422e-14, 5.547e-14, 3.636e-14, 2.368e-14, 1.536e-14,
00299 9.937e-15, 6.39e-15, 4.101e-15, 2.61e-15, 1.659e-15, 1.052e-15,
00300 6.638e-16, 4.172e-16, 2.61e-16, 1.63e-16, 1.013e-16, 6.275e-17,
00301 3.879e-17, 2.383e-17, 1.461e-17, 8.918e-18, 5.43e-18, 3.301e-18,
00302 1.997e-18, 1.203e-18, 7.216e-19, 4.311e-19, 2.564e-19, 1.519e-19,
00303 8.911e-20, 5.203e-20, 3.026e-20, 1.748e-20, 9.99e-21, 5.673e-21,
00304 3.215e-21, 1.799e-21, 1.006e-21, 5.628e-22, 3.146e-22, 1.766e-22,
00305 9.94e-23, 5.614e-23, 3.206e-23, 1.841e-23, 1.071e-23, 6.366e-24,
00306 3.776e-24, 2.238e-24, 1.326e-24, 8.253e-25, 5.201e-25, 3.279e-25,
00307 2.108e-25, 1.395e-25, 9.326e-26, 6.299e-26, 4.365e-26, 3.104e-26,
00308 2.219e-26, 1.621e-26, 1.185e-26, 8.92e-27, 6.804e-27, 5.191e-27,
00309 4.041e-27
00310 };
00311
00312 static double co[121] = {
00313 1.907e-07, 1.553e-07, 1.362e-07, 1.216e-07, 1.114e-07, 1.036e-07,
00314 9.737e-08, 9.152e-08, 8.559e-08, 7.966e-08, 7.277e-08, 6.615e-08,
00315 5.884e-08, 5.22e-08, 4.699e-08, 4.284e-08, 3.776e-08, 3.274e-08,
00316 2.845e-08, 2.479e-08, 2.246e-08, 2.054e-08, 1.991e-08, 1.951e-08,
00317 1.94e-08, 2.009e-08, 2.1e-08, 2.201e-08, 2.322e-08, 2.45e-08,
00318 2.602e-08, 2.73e-08, 2.867e-08, 2.998e-08, 3.135e-08, 3.255e-08,
00319 3.352e-08, 3.426e-08, 3.484e-08, 3.53e-08, 3.593e-08, 3.671e-08,
00320 3.759e-08, 3.945e-08, 4.192e-08, 4.49e-08, 5.03e-08, 5.703e-08,
00321 6.538e-08, 7.878e-08, 9.644e-08, 1.196e-07, 1.498e-07, 1.904e-07,
00322 2.422e-07, 3.055e-07, 3.804e-07, 4.747e-07, 5.899e-07, 7.272e-07,
00323 8.91e-07, 1.071e-06, 1.296e-06, 1.546e-06, 1.823e-06, 2.135e-06,
00324 2.44e-06, 2.714e-06, 2.967e-06, 3.189e-06, 3.391e-06, 3.58e-06,
00325 3.773e-06, 4.022e-06, 4.346e-06, 4.749e-06, 5.199e-06, 5.668e-06,
00326 6.157e-06, 6.688e-06, 7.254e-06, 7.867e-06, 8.539e-06, 9.26e-06,
00327 1.009e-05, 1.119e-05, 1.228e-05, 1.365e-05, 1.506e-05, 1.641e-05,
00328 1.784e-05, 1.952e-05, 2.132e-05, 2.323e-05, 2.531e-05, 2.754e-05,
00329 3.047e-05, 3.459e-05, 3.922e-05, 4.439e-05, 4.825e-05, 5.077e-05,
00330 5.34e-05, 5.618e-05, 5.909e-05, 6.207e-05, 6.519e-05, 6.845e-05,
00331 6.819e-05, 6.726e-05, 6.622e-05, 6.512e-05, 6.671e-05, 6.862e-05,
00332 7.048e-05, 7.264e-05, 7.3e-05, 7.3e-05, 7.3e-05, 7.3e-05, 7.3e-05,
00333 };
00334
00335 static double cof2[121] = {
00336 7.5e-14, 1.055e-13, 1.485e-13, 2.111e-13, 3.001e-13, 4.333e-13,
00337 6.269e-13, 9.221e-13, 1.364e-12, 2.046e-12, 3.093e-12, 4.703e-12,
00338 7.225e-12, 1.113e-11, 1.66e-11, 2.088e-11, 2.626e-11, 3.433e-11,
00339 4.549e-11, 5.886e-11, 7.21e-11, 8.824e-11, 1.015e-10, 1.155e-10,
00340 1.288e-10, 1.388e-10, 1.497e-10, 1.554e-10, 1.606e-10, 1.639e-10,
00341 1.64e-10, 1.64e-10, 1.596e-10, 1.542e-10, 1.482e-10, 1.382e-10,
00342 1.289e-10, 1.198e-10, 1.109e-10, 1.026e-10, 9.484e-11, 8.75e-11,
00343 8.086e-11, 7.49e-11, 6.948e-11, 6.446e-11, 5.961e-11, 5.505e-11,
00344 5.085e-11, 4.586e-11, 4.1e-11, 3.665e-11, 3.235e-11, 2.842e-11,
00345 2.491e-11, 2.11e-11, 1.769e-11, 1.479e-11, 1.197e-11, 9.631e-12,
00346 7.74e-12, 6.201e-12, 4.963e-12, 3.956e-12, 3.151e-12, 2.507e-12,
00347 1.99e-12, 1.576e-12, 1.245e-12, 9.83e-13, 7.742e-13, 6.088e-13,
00348 4.782e-13, 3.745e-13, 2.929e-13, 2.286e-13, 1.782e-13, 1.388e-13,
00349 1.079e-13, 8.362e-14, 6.471e-14, 4.996e-14, 3.85e-14, 2.96e-14,
00350 2.265e-14, 1.729e-14, 1.317e-14, 9.998e-15, 7.549e-15, 5.683e-15,
00351 4.273e-15, 3.193e-15, 2.385e-15, 1.782e-15, 1.331e-15, 9.957e-16,
00352 7.461e-16, 5.601e-16, 4.228e-16, 3.201e-16, 2.438e-16, 1.878e-16,
00353 1.445e-16, 1.111e-16, 8.544e-17, 6.734e-17, 5.341e-17, 4.237e-17,
00354 3.394e-17, 2.759e-17, 2.254e-17, 1.851e-17, 1.54e-17, 1.297e-17,
00355 1.096e-17, 9.365e-18, 8e-18, 6.938e-18, 6.056e-18, 5.287e-18,
00356 4.662e-18
00357 };
00358
00359 static double f11[121] = {
```

```
00360      2.65e-10, 2.65e-10, 2.65e-10, 2.65e-10, 2.65e-10, 2.65e-10, 2.65e-10,
00361      2.65e-10, 2.65e-10, 2.65e-10, 2.65e-10, 2.65e-10, 2.635e-10, 2.536e-10,
00362      2.44e-10, 2.348e-10, 2.258e-10, 2.153e-10, 2.046e-10, 1.929e-10,
00363      1.782e-10, 1.648e-10, 1.463e-10, 1.291e-10, 1.1e-10, 8.874e-11,
00364      7.165e-11, 5.201e-11, 3.744e-11, 2.577e-11, 1.64e-11, 1.048e-11,
00365      5.993e-12, 3.345e-12, 1.839e-12, 9.264e-13, 4.688e-13, 2.329e-13,
00366      1.129e-13, 5.505e-14, 2.825e-14, 1.492e-14, 7.997e-15, 5.384e-15,
00367      3.988e-15, 2.955e-15, 2.196e-15, 1.632e-15, 1.214e-15, 9.025e-16,
00368      6.708e-16, 4.984e-16, 3.693e-16, 2.733e-16, 2.013e-16, 1.481e-16,
00369      1.087e-16, 7.945e-17, 5.782e-17, 4.195e-17, 3.038e-17, 2.19e-17,
00370      1.577e-17, 1.128e-17, 8.063e-18, 5.753e-18, 4.09e-18, 2.899e-18,
00371      2.048e-18, 1.444e-18, 1.015e-18, 7.12e-19, 4.985e-19, 3.474e-19,
00372      2.417e-19, 1.677e-19, 1.161e-19, 8.029e-20, 5.533e-20, 3.799e-20,
00373      2.602e-20, 1.776e-20, 1.209e-20, 8.202e-21, 5.522e-21, 3.707e-21,
00374      2.48e-21, 1.652e-21, 1.091e-21, 7.174e-22, 4.709e-22, 3.063e-22,
00375      1.991e-22, 1.294e-22, 8.412e-23, 5.483e-23, 3.581e-23, 2.345e-23,
00376      1.548e-23, 1.027e-23, 6.869e-24, 4.673e-24, 3.173e-24, 2.153e-24,
00377      1.461e-24, 1.028e-24, 7.302e-25, 5.188e-25, 3.739e-25, 2.753e-25,
00378      2.043e-25, 1.528e-25, 1.164e-25, 9.041e-26, 7.051e-26, 5.587e-26,
00379      4.428e-26, 3.588e-26, 2.936e-26, 2.402e-26, 1.995e-26
00380  };
00381
00382  static double f12[121] = {
00383      5.45e-10, 5.45e-10, 5.45e-10, 5.45e-10, 5.45e-10, 5.45e-10, 5.45e-10,
00384      5.45e-10, 5.45e-10, 5.45e-10, 5.45e-10, 5.45e-10, 5.429e-10, 5.291e-10,
00385      5.155e-10, 5.022e-10, 4.893e-10, 4.772e-10, 4.655e-10, 4.497e-10,
00386      4.249e-10, 4.015e-10, 3.632e-10, 3.261e-10, 2.858e-10, 2.408e-10,
00387      2.03e-10, 1.685e-10, 1.4e-10, 1.163e-10, 9.65e-11, 8.02e-11, 6.705e-11,
00388      5.624e-11, 4.764e-11, 4.249e-11, 3.792e-11, 3.315e-11, 2.819e-11,
00389      2.4e-11, 1.999e-11, 1.64e-11, 1.352e-11, 1.14e-11, 9.714e-12,
00390      8.28e-12, 7.176e-12, 6.251e-12, 5.446e-12, 4.72e-12, 4.081e-12,
00391      3.528e-12, 3.08e-12, 2.699e-12, 2.359e-12, 2.111e-12, 1.901e-12,
00392      1.709e-12, 1.534e-12, 1.376e-12, 1.233e-12, 1.103e-12, 9.869e-13,
00393      8.808e-13, 7.859e-13, 7.008e-13, 6.241e-13, 5.553e-13, 4.935e-13,
00394      4.383e-13, 3.889e-13, 3.447e-13, 3.054e-13, 2.702e-13, 2.389e-13,
00395      2.11e-13, 1.862e-13, 1.643e-13, 1.448e-13, 1.274e-13, 1.121e-13,
00396      9.844e-14, 8.638e-14, 7.572e-14, 6.62e-14, 5.782e-14, 5.045e-14,
00397      4.394e-14, 3.817e-14, 3.311e-14, 2.87e-14, 2.48e-14, 2.142e-14,
00398      1.851e-14, 1.599e-14, 1.383e-14, 1.196e-14, 1.036e-14, 9e-15,
00399      7.828e-15, 6.829e-15, 5.992e-15, 5.254e-15, 4.606e-15, 4.037e-15,
00400      3.583e-15, 3.19e-15, 2.841e-15, 2.542e-15, 2.291e-15, 2.07e-15,
00401      1.875e-15, 1.71e-15, 1.57e-15, 1.442e-15, 1.333e-15, 1.232e-15,
00402      1.147e-15, 1.071e-15, 1.001e-15, 9.396e-16
00403  };
00404
00405  static double f14[121] = {
00406      9e-11, 9e-11, 9e-11, 9e-11, 9e-11, 9e-11, 9e-11, 9e-11, 9e-11, 9e-11,
00407      9e-11, 9e-11, 9e-11, 9e-11, 9e-11, 8.91e-11, 8.73e-11, 8.46e-11,
00408      8.19e-11, 7.92e-11, 7.74e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00409      7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00410      7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00411      7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00412      7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00413      7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00414      7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00415      7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00416      7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00417      7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00418      7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00419      7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00420      7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00421      7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00422      7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11
00423  };
00424
00425  static double f22[121] = {
00426      1.4e-10, 1.4e-10, 1.4e-10, 1.4e-10, 1.4e-10, 1.4e-10, 1.4e-10,
00427      1.4e-10, 1.4e-10, 1.4e-10, 1.372e-10, 1.317e-10, 1.235e-10, 1.153e-10,
00428      1.075e-10, 1.002e-10, 9.332e-11, 8.738e-11, 8.194e-11, 7.7e-11,
00429      7.165e-11, 6.753e-11, 6.341e-11, 5.971e-11, 5.6e-11, 5.229e-11,
00430      4.859e-11, 4.488e-11, 4.118e-11, 3.83e-11, 3.568e-11, 3.308e-11,
00431      3.047e-11, 2.82e-11, 2.594e-11, 2.409e-11, 2.237e-11, 2.065e-11,
00432      1.894e-11, 1.771e-11, 1.647e-11, 1.532e-11, 1.416e-11, 1.332e-11,
00433      1.246e-11, 1.161e-11, 1.087e-11, 1.017e-11, 9.471e-12, 8.853e-12,
00434      8.235e-12, 7.741e-12, 7.247e-12, 6.836e-12, 6.506e-12, 6.176e-12,
00435      5.913e-12, 5.65e-12, 5.419e-12, 5.221e-12, 5.024e-12, 4.859e-12,
00436      4.694e-12, 4.546e-12, 4.414e-12, 4.282e-12, 4.15e-12, 4.019e-12,
00437      3.903e-12, 3.805e-12, 3.706e-12, 3.607e-12, 3.508e-12, 3.41e-12,
00438      3.31e-12, 3.212e-12, 3.129e-12, 3.047e-12, 2.964e-12, 2.882e-12,
00439      2.8e-12, 2.734e-12, 2.668e-12, 2.602e-12, 2.537e-12, 2.471e-12,
00440      2.421e-12, 2.372e-12, 2.322e-12, 2.273e-12, 2.224e-12, 2.182e-12,
00441      2.141e-12, 2.1e-12, 2.059e-12, 2.018e-12, 1.977e-12, 1.935e-12,
00442      1.894e-12, 1.853e-12, 1.812e-12, 1.77e-12, 1.73e-12, 1.688e-12,
00443      1.647e-12, 1.606e-12, 1.565e-12, 1.524e-12, 1.483e-12, 1.441e-12,
00444      1.4e-12, 1.359e-12, 1.317e-12, 1.276e-12, 1.235e-12, 1.194e-12,
00445      1.153e-12, 1.112e-12, 1.071e-12, 1.029e-12, 9.883e-13
00446  };
```

```
00447
00448 static double h2o[121] = {
00449     0.01166, 0.008269, 0.005742, 0.003845, 0.00277, 0.001897, 0.001272,
00450     0.000827, 0.000539, 0.0003469, 0.0001579, 3.134e-05, 1.341e-05,
00451     6.764e-06, 4.498e-06, 3.703e-06, 3.724e-06, 3.899e-06, 4.002e-06,
00452     4.122e-06, 4.277e-06, 4.438e-06, 4.558e-06, 4.673e-06, 4.763e-06,
00453     4.809e-06, 4.856e-06, 4.936e-06, 5.021e-06, 5.114e-06, 5.222e-06,
00454     5.331e-06, 5.414e-06, 5.488e-06, 5.563e-06, 5.633e-06, 5.704e-06,
00455     5.767e-06, 5.819e-06, 5.872e-06, 5.914e-06, 5.949e-06, 5.984e-06,
00456     6.015e-06, 6.044e-06, 6.073e-06, 6.104e-06, 6.136e-06, 6.167e-06,
00457     6.189e-06, 6.208e-06, 6.226e-06, 6.212e-06, 6.185e-06, 6.158e-06,
00458     6.114e-06, 6.066e-06, 6.018e-06, 5.877e-06, 5.728e-06, 5.582e-06,
00459     5.437e-06, 5.296e-06, 5.156e-06, 5.02e-06, 4.886e-06, 4.754e-06,
00460     4.625e-06, 4.498e-06, 4.374e-06, 4.242e-06, 4.096e-06, 3.955e-06,
00461     3.817e-06, 3.683e-06, 3.491e-06, 3.204e-06, 2.94e-06, 2.696e-06,
00462     2.47e-06, 2.252e-06, 2.019e-06, 1.808e-06, 1.618e-06, 1.445e-06,
00463     1.285e-06, 1.105e-06, 9.489e-07, 8.121e-07, 6.938e-07, 5.924e-07,
00464     5.04e-07, 4.288e-07, 3.648e-07, 3.103e-07, 2.642e-07, 2.252e-07,
00465     1.921e-07, 1.643e-07, 1.408e-07, 1.211e-07, 1.048e-07, 9.063e-08,
00466     7.835e-08, 6.774e-08, 5.936e-08, 5.221e-08, 4.592e-08, 4.061e-08,
00467     3.62e-08, 3.236e-08, 2.902e-08, 2.62e-08, 2.383e-08, 2.171e-08,
00468     1.989e-08, 1.823e-08, 1.684e-08, 1.562e-08, 1.449e-08, 1.351e-08
00469 };
00470
00471 static double h2o2[121] = {
00472     1.779e-10, 7.938e-10, 8.953e-10, 8.032e-10, 6.564e-10, 5.159e-10,
00473     4.003e-10, 3.026e-10, 2.222e-10, 1.58e-10, 1.044e-10, 6.605e-11,
00474     3.413e-11, 1.453e-11, 1.062e-11, 1.009e-11, 9.597e-12, 1.175e-11,
00475     1.572e-11, 2.091e-11, 2.746e-11, 3.603e-11, 4.791e-11, 6.387e-11,
00476     8.239e-11, 1.007e-10, 1.23e-10, 1.363e-10, 1.489e-10, 1.585e-10,
00477     1.608e-10, 1.632e-10, 1.576e-10, 1.502e-10, 1.423e-10, 1.302e-10,
00478     1.192e-10, 1.085e-10, 9.795e-11, 8.854e-11, 8.057e-11, 7.36e-11,
00479     6.736e-11, 6.362e-11, 6.087e-11, 5.825e-11, 5.623e-11, 5.443e-11,
00480     5.27e-11, 5.098e-11, 4.931e-11, 4.769e-11, 4.611e-11, 4.458e-11,
00481     4.308e-11, 4.102e-11, 3.887e-11, 3.682e-11, 3.521e-11, 3.369e-11,
00482     3.224e-11, 3.082e-11, 2.946e-11, 2.814e-11, 2.687e-11, 2.566e-11,
00483     2.449e-11, 2.336e-11, 2.227e-11, 2.123e-11, 2.023e-11, 1.927e-11,
00484     1.835e-11, 1.746e-11, 1.661e-11, 1.58e-11, 1.502e-11, 1.428e-11,
00485     1.357e-11, 1.289e-11, 1.224e-11, 1.161e-11, 1.102e-11, 1.045e-11,
00486     9.895e-12, 9.369e-12, 8.866e-12, 8.386e-12, 7.922e-12, 7.479e-12,
00487     7.06e-12, 6.656e-12, 6.274e-12, 5.914e-12, 5.575e-12, 5.257e-12,
00488     4.959e-12, 4.679e-12, 4.42e-12, 4.178e-12, 3.954e-12, 3.75e-12,
00489     3.557e-12, 3.372e-12, 3.198e-12, 3.047e-12, 2.908e-12, 2.775e-12,
00490     2.653e-12, 2.544e-12, 2.442e-12, 2.346e-12, 2.26e-12, 2.183e-12,
00491     2.11e-12, 2.044e-12, 1.98e-12, 1.924e-12, 1.871e-12, 1.821e-12,
00492     1.775e-12
00493 };
00494
00495 static double hcn[121] = {
00496     5.5e-10, 5.5e-10, 5.5e-10, 5.5e-10, 5.5e-10, 5.5e-10,
00497     5.5e-10, 5.5e-10, 5.5e-10, 5.5e-10, 5.498e-10, 5.495e-10, 5.493e-10,
00498     5.49e-10, 5.488e-10, 4.717e-10, 3.946e-10, 3.174e-10, 2.4e-10,
00499     1.626e-10, 1.619e-10, 1.612e-10, 1.602e-10, 1.593e-10, 1.582e-10,
00500     1.572e-10, 1.56e-10, 1.549e-10, 1.539e-10, 1.53e-10, 1.519e-10,
00501     1.506e-10, 1.487e-10, 1.467e-10, 1.449e-10, 1.43e-10, 1.413e-10,
00502     1.397e-10, 1.382e-10, 1.368e-10, 1.354e-10, 1.337e-10, 1.315e-10,
00503     1.292e-10, 1.267e-10, 1.241e-10, 1.215e-10, 1.19e-10, 1.165e-10,
00504     1.141e-10, 1.118e-10, 1.096e-10, 1.072e-10, 1.047e-10, 1.021e-10,
00505     9.968e-11, 9.739e-11, 9.539e-11, 9.339e-11, 9.135e-11, 8.898e-11,
00506     8.664e-11, 8.439e-11, 8.249e-11, 8.075e-11, 7.904e-11, 7.735e-11,
00507     7.565e-11, 7.399e-11, 7.245e-11, 7.109e-11, 6.982e-11, 6.863e-11,
00508     6.755e-11, 6.657e-11, 6.587e-11, 6.527e-11, 6.476e-11, 6.428e-11,
00509     6.382e-11, 6.343e-11, 6.307e-11, 6.272e-11, 6.238e-11, 6.205e-11,
00510     6.17e-11, 6.137e-11, 6.102e-11, 6.072e-11, 6.046e-11, 6.03e-11,
00511     6.018e-11, 6.01e-11, 6.001e-11, 5.992e-11, 5.984e-11, 5.975e-11,
00512     5.967e-11, 5.958e-11, 5.95e-11, 5.941e-11, 5.933e-11, 5.925e-11,
00513     5.916e-11, 5.908e-11, 5.899e-11, 5.891e-11, 5.883e-11, 5.874e-11,
00514     5.866e-11, 5.858e-11, 5.85e-11, 5.841e-11, 5.833e-11, 5.825e-11,
00515     5.817e-11, 5.808e-11, 5.8e-11, 5.792e-11, 5.784e-11
00516 };
00517
00518 static double hno3[121] = {
00519     1.809e-10, 7.234e-10, 5.899e-10, 4.342e-10, 3.277e-10, 2.661e-10,
00520     2.35e-10, 2.267e-10, 2.389e-10, 2.651e-10, 3.255e-10, 4.099e-10,
00521     5.42e-10, 6.978e-10, 8.807e-10, 1.112e-09, 1.405e-09, 2.04e-09,
00522     3.111e-09, 4.5e-09, 5.762e-09, 7.37e-09, 7.852e-09, 8.109e-09,
00523     8.067e-09, 7.554e-09, 7.076e-09, 6.268e-09, 5.524e-09, 4.749e-09,
00524     3.909e-09, 3.223e-09, 2.517e-09, 1.942e-09, 1.493e-09, 1.122e-09,
00525     8.449e-10, 6.361e-10, 4.787e-10, 3.611e-10, 2.804e-10, 2.215e-10,
00526     1.758e-10, 1.441e-10, 1.197e-10, 9.953e-11, 8.505e-11, 7.334e-11,
00527     6.325e-11, 5.625e-11, 5.058e-11, 4.548e-11, 4.122e-11, 3.748e-11,
00528     3.402e-11, 3.088e-11, 2.8e-11, 2.536e-11, 2.293e-11, 2.072e-11,
00529     1.871e-11, 1.687e-11, 1.52e-11, 1.368e-11, 1.23e-11, 1.105e-11,
00530     9.922e-12, 8.898e-12, 7.972e-12, 7.139e-12, 6.385e-12, 5.708e-12,
00531     5.099e-12, 4.549e-12, 4.056e-12, 3.613e-12, 3.216e-12, 2.862e-12,
00532     2.544e-12, 2.259e-12, 2.004e-12, 1.776e-12, 1.572e-12, 1.391e-12,
00533     1.227e-12, 1.082e-12, 9.528e-13, 8.379e-13, 7.349e-13, 6.436e-13,
```

```
00534     5.634e-13, 4.917e-13, 4.291e-13, 3.745e-13, 3.267e-13, 2.854e-13,
00535     2.494e-13, 2.181e-13, 1.913e-13, 1.68e-13, 1.479e-13, 1.31e-13,
00536     1.159e-13, 1.025e-13, 9.067e-14, 8.113e-14, 7.281e-14, 6.535e-14,
00537     5.892e-14, 5.348e-14, 4.867e-14, 4.439e-14, 4.073e-14, 3.76e-14,
00538     3.476e-14, 3.229e-14, 3e-14, 2.807e-14, 2.635e-14, 2.473e-14,
00539     2.332e-14
00540 };
00541
00542 static double hno4[121] = {
00543     6.118e-12, 3.594e-12, 2.807e-12, 3.04e-12, 4.458e-12, 7.986e-12,
00544     1.509e-11, 2.661e-11, 3.738e-11, 4.652e-11, 4.429e-11, 3.992e-11,
00545     3.347e-11, 3.005e-11, 3.173e-11, 4.055e-11, 5.812e-11, 8.489e-11,
00546     1.19e-10, 1.482e-10, 1.766e-10, 2.103e-10, 2.35e-10, 2.598e-10,
00547     2.801e-10, 2.899e-10, 3e-10, 2.817e-10, 2.617e-10, 2.332e-10,
00548     1.933e-10, 1.605e-10, 1.232e-10, 9.285e-11, 6.941e-11, 4.951e-11,
00549     3.539e-11, 2.402e-11, 1.522e-11, 9.676e-12, 6.056e-12, 3.745e-12,
00550     2.34e-12, 1.463e-12, 9.186e-13, 5.769e-13, 3.322e-13, 1.853e-13,
00551     1.035e-13, 7.173e-14, 5.382e-14, 4.036e-14, 3.401e-14, 2.997e-14,
00552     2.635e-14, 2.316e-14, 2.034e-14, 1.783e-14, 1.56e-14, 1.363e-14,
00553     1.19e-14, 1.037e-14, 9.032e-15, 7.846e-15, 6.813e-15, 5.912e-15,
00554     5.121e-15, 4.431e-15, 3.829e-15, 3.306e-15, 2.851e-15, 2.456e-15,
00555     2.114e-15, 1.816e-15, 1.559e-15, 1.337e-15, 1.146e-15, 9.811e-16,
00556     8.389e-16, 7.162e-16, 6.109e-16, 5.203e-16, 4.425e-16, 3.76e-16,
00557     3.184e-16, 2.692e-16, 2.274e-16, 1.917e-16, 1.61e-16, 1.35e-16,
00558     1.131e-16, 9.437e-17, 7.874e-17, 6.57e-17, 5.481e-17, 4.579e-17,
00559     3.828e-17, 3.204e-17, 2.691e-17, 2.264e-17, 1.912e-17, 1.626e-17,
00560     1.382e-17, 1.174e-17, 9.972e-18, 8.603e-18, 7.45e-18, 6.453e-18,
00561     5.623e-18, 4.944e-18, 4.361e-18, 3.859e-18, 3.443e-18, 3.096e-18,
00562     2.788e-18, 2.528e-18, 2.293e-18, 2.099e-18, 1.929e-18, 1.773e-18,
00563     1.64e-18
00564 };
00565
00566 static double hocl[121] = {
00567     1.056e-12, 1.194e-12, 1.35e-12, 1.531e-12, 1.737e-12, 1.982e-12,
00568     2.263e-12, 2.599e-12, 2.991e-12, 3.459e-12, 4.012e-12, 4.662e-12,
00569     5.438e-12, 6.35e-12, 7.425e-12, 8.686e-12, 1.016e-11, 1.188e-11,
00570     1.389e-11, 1.659e-11, 2.087e-11, 2.621e-11, 3.265e-11, 4.064e-11,
00571     4.859e-11, 5.441e-11, 6.09e-11, 6.373e-11, 6.611e-11, 6.94e-11,
00572     7.44e-11, 7.97e-11, 8.775e-11, 9.722e-11, 1.064e-10, 1.089e-10,
00573     1.114e-10, 1.106e-10, 1.053e-10, 1.004e-10, 9.006e-11, 7.778e-11,
00574     6.739e-11, 5.636e-11, 4.655e-11, 3.845e-11, 3.042e-11, 2.368e-11,
00575     1.845e-11, 1.442e-11, 1.127e-11, 8.814e-12, 6.544e-12, 4.763e-12,
00576     3.449e-12, 2.612e-12, 1.999e-12, 1.526e-12, 1.16e-12, 8.793e-13,
00577     6.655e-13, 5.017e-13, 3.778e-13, 2.829e-13, 2.117e-13, 1.582e-13,
00578     1.178e-13, 8.755e-14, 6.486e-14, 4.799e-14, 3.54e-14, 2.606e-14,
00579     1.916e-14, 1.403e-14, 1.026e-14, 7.48e-15, 5.446e-15, 3.961e-15,
00580     2.872e-15, 2.076e-15, 1.498e-15, 1.077e-15, 7.726e-16, 5.528e-16,
00581     3.929e-16, 2.785e-16, 1.969e-16, 1.386e-16, 9.69e-17, 6.747e-17,
00582     4.692e-17, 3.236e-17, 2.232e-17, 1.539e-17, 1.061e-17, 7.332e-18,
00583     5.076e-18, 3.522e-18, 2.461e-18, 1.726e-18, 1.22e-18, 8.75e-19,
00584     6.264e-19, 4.482e-19, 3.207e-19, 2.368e-19, 1.762e-19, 1.312e-19,
00585     9.891e-20, 7.595e-20, 5.87e-20, 4.567e-20, 3.612e-20, 2.904e-20,
00586     2.343e-20, 1.917e-20, 1.568e-20, 1.308e-20, 1.1e-20, 9.25e-21,
00587     7.881e-21
00588 };
00589
00590 static double n2o[121] = {
00591     3.17e-07, 3.17e-07, 3.17e-07, 3.17e-07, 3.17e-07, 3.17e-07, 3.17e-07,
00592     3.17e-07, 3.17e-07, 3.17e-07, 3.124e-07, 3.077e-07, 3.03e-07,
00593     2.984e-07, 2.938e-07, 2.892e-07, 2.847e-07, 2.779e-07, 2.705e-07,
00594     2.631e-07, 2.557e-07, 2.484e-07, 2.345e-07, 2.201e-07, 2.01e-07,
00595     1.754e-07, 1.532e-07, 1.329e-07, 1.154e-07, 1.003e-07, 8.735e-08,
00596     7.617e-08, 6.512e-08, 5.547e-08, 4.709e-08, 3.915e-08, 3.259e-08,
00597     2.738e-08, 2.327e-08, 1.98e-08, 1.711e-08, 1.493e-08, 1.306e-08,
00598     1.165e-08, 1.049e-08, 9.439e-09, 8.375e-09, 7.391e-09, 6.525e-09,
00599     5.759e-09, 5.083e-09, 4.485e-09, 3.953e-09, 3.601e-09, 3.27e-09,
00600     2.975e-09, 2.757e-09, 2.556e-09, 2.37e-09, 2.195e-09, 2.032e-09,
00601     1.912e-09, 1.79e-09, 1.679e-09, 1.572e-09, 1.482e-09, 1.402e-09,
00602     1.326e-09, 1.254e-09, 1.187e-09, 1.127e-09, 1.071e-09, 1.02e-09,
00603     9.673e-10, 9.193e-10, 8.752e-10, 8.379e-10, 8.017e-10, 7.66e-10,
00604     7.319e-10, 7.004e-10, 6.721e-10, 6.459e-10, 6.199e-10, 5.942e-10,
00605     5.703e-10, 5.488e-10, 5.283e-10, 5.082e-10, 4.877e-10, 4.696e-10,
00606     4.52e-10, 4.355e-10, 4.198e-10, 4.039e-10, 3.888e-10, 3.754e-10,
00607     3.624e-10, 3.499e-10, 3.381e-10, 3.267e-10, 3.163e-10, 3.058e-10,
00608     2.959e-10, 2.864e-10, 2.77e-10, 2.686e-10, 2.604e-10, 2.534e-10,
00609     2.462e-10, 2.386e-10, 2.318e-10, 2.247e-10, 2.189e-10, 2.133e-10,
00610     2.071e-10, 2.014e-10, 1.955e-10, 1.908e-10, 1.86e-10, 1.817e-10
00611 };
00612
00613 static double n2o5[121] = {
00614     1.231e-11, 3.035e-12, 1.702e-12, 9.877e-13, 8.081e-13, 9.039e-13,
00615     1.169e-12, 1.474e-12, 1.651e-12, 1.795e-12, 1.998e-12, 2.543e-12,
00616     4.398e-12, 7.698e-12, 1.28e-11, 2.131e-11, 3.548e-11, 5.894e-11,
00617     7.645e-11, 1.089e-10, 1.391e-10, 1.886e-10, 2.386e-10, 2.986e-10,
00618     3.487e-10, 3.994e-10, 4.5e-10, 4.6e-10, 4.591e-10, 4.1e-10, 3.488e-10,
00619     2.846e-10, 2.287e-10, 1.696e-10, 1.011e-10, 6.428e-11, 4.324e-11,
00620     2.225e-11, 6.214e-12, 3.608e-12, 8.793e-13, 4.491e-13, 1.04e-13,
```

```
00621    6.1e-14, 3.436e-14, 6.671e-15, 1.171e-15, 5.848e-16, 1.212e-16,
00622    1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00623    1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00624    1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00625    1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00626    1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00627    1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00628    1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00629    1e-16, 1e-16
00630 };
00631
00632 static double nh3[121] = {
00633     1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10,
00634     1e-10, 1e-10, 1e-10, 1e-10, 9.444e-11, 8.488e-11, 7.241e-11, 5.785e-11,
00635     4.178e-11, 3.018e-11, 2.18e-11, 1.574e-11, 1.137e-11, 8.211e-12,
00636     5.973e-12, 4.327e-12, 3.118e-12, 2.234e-12, 1.573e-12, 1.04e-12,
00637     6.762e-13, 4.202e-13, 2.406e-13, 1.335e-13, 6.938e-14, 3.105e-14,
00638     1.609e-14, 1.033e-14, 6.432e-15, 4.031e-15, 2.555e-15, 1.656e-15,
00639     1.115e-15, 7.904e-16, 5.63e-16, 4.048e-16, 2.876e-16, 2.004e-16,
00640     1.356e-16, 9.237e-17, 6.235e-17, 4.223e-17, 3.009e-17, 2.328e-17,
00641     2.002e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00642     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00643     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00644     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00645     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00646     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00647     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00648     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00649     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00650     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00651     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00652     1.914e-17
00653 };
00654
00655 static double no[121] = {
00656     2.586e-10, 4.143e-11, 1.566e-11, 9.591e-12, 8.088e-12, 8.462e-12,
00657     1.013e-11, 1.328e-11, 1.855e-11, 2.678e-11, 3.926e-11, 5.464e-11,
00658     7.012e-11, 8.912e-11, 1.127e-10, 1.347e-10, 1.498e-10, 1.544e-10,
00659     1.602e-10, 1.824e-10, 2.078e-10, 2.366e-10, 2.691e-10, 5.141e-10,
00660     8.259e-10, 1.254e-09, 1.849e-09, 2.473e-09, 3.294e-09, 4.16e-09,
00661     5.095e-09, 6.11e-09, 6.93e-09, 7.888e-09, 8.903e-09, 9.713e-09,
00662     1.052e-08, 1.115e-08, 1.173e-08, 1.21e-08, 1.228e-08, 1.239e-08,
00663     1.231e-08, 1.213e-08, 1.192e-08, 1.138e-08, 1.085e-08, 1.008e-08,
00664     9.224e-09, 8.389e-09, 7.262e-09, 6.278e-09, 5.335e-09, 4.388e-09,
00665     3.589e-09, 2.761e-09, 2.129e-09, 1.633e-09, 1.243e-09, 9.681e-10,
00666     8.355e-10, 7.665e-10, 7.442e-10, 8.584e-10, 9.732e-10, 1.063e-09,
00667     1.163e-09, 1.286e-09, 1.472e-09, 1.707e-09, 2.032e-09, 2.474e-09,
00668     2.977e-09, 3.506e-09, 4.102e-09, 5.013e-09, 6.493e-09, 8.414e-09,
00669     1.077e-08, 1.367e-08, 1.777e-08, 2.625e-08, 3.926e-08, 5.545e-08,
00670     7.195e-08, 9.464e-08, 1.404e-07, 2.183e-07, 3.329e-07, 4.535e-07,
00671     6.158e-07, 8.187e-07, 1.075e-06, 1.422e-06, 1.979e-06, 2.71e-06,
00672     3.58e-06, 4.573e-06, 5.951e-06, 7.999e-06, 1.072e-05, 1.372e-05,
00673     1.697e-05, 2.112e-05, 2.643e-05, 3.288e-05, 3.994e-05, 4.794e-05,
00674     5.606e-05, 6.383e-05, 7.286e-05, 8.156e-05, 8.883e-05, 9.469e-05,
00675     9.848e-05, 0.0001023, 0.0001066, 0.0001115, 0.0001145, 0.0001142,
00676     0.0001133
00677 };
00678
00679 static double no2[121] = {
00680     3.036e-09, 2.945e-10, 9.982e-11, 5.069e-11, 3.485e-11, 2.982e-11,
00681     2.947e-11, 3.164e-11, 3.714e-11, 4.586e-11, 6.164e-11, 8.041e-11,
00682     9.982e-11, 1.283e-10, 1.73e-10, 2.56e-10, 3.909e-10, 5.959e-10,
00683     9.081e-10, 1.384e-09, 1.788e-09, 2.189e-09, 2.686e-09, 3.091e-09,
00684     3.49e-09, 3.796e-09, 4.2e-09, 5.103e-09, 6.005e-09, 6.3e-09, 6.706e-09,
00685     7.07e-09, 7.434e-09, 7.663e-09, 7.788e-09, 7.8e-09, 7.597e-09,
00686     7.482e-09, 7.227e-09, 6.403e-09, 5.585e-09, 4.606e-09, 3.703e-09,
00687     2.984e-09, 2.183e-09, 1.48e-09, 8.441e-10, 5.994e-10, 3.799e-10,
00688     2.751e-10, 1.927e-10, 1.507e-10, 1.102e-10, 6.971e-11, 5.839e-11,
00689     3.904e-11, 3.087e-11, 2.176e-11, 1.464e-11, 1.209e-11, 8.497e-12,
00690     6.477e-12, 4.371e-12, 2.914e-12, 2.424e-12, 1.753e-12, 1.35e-12,
00691     9.417e-13, 6.622e-13, 5.148e-13, 3.841e-13, 3.446e-13, 3.01e-13,
00692     2.551e-13, 2.151e-13, 1.829e-13, 1.64e-13, 1.475e-13, 1.352e-13,
00693     1.155e-13, 9.963e-14, 9.771e-14, 9.577e-14, 9.384e-14, 9.186e-14,
00694     9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14,
00695     9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14,
00696     9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14,
00697     9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14
00698 };
00699
00700 static double o3[121] = {
00701     2.218e-08, 3.394e-08, 3.869e-08, 4.219e-08, 4.501e-08, 4.778e-08,
00702     5.067e-08, 5.402e-08, 5.872e-08, 6.521e-08, 7.709e-08, 9.461e-08,
00703     1.269e-07, 1.853e-07, 2.723e-07, 3.964e-07, 5.773e-07, 8.2e-07,
00704     1.155e-06, 1.59e-06, 2.076e-06, 2.706e-06, 3.249e-06, 3.848e-06,
00705     4.459e-06, 4.986e-06, 5.573e-06, 5.958e-06, 6.328e-06, 6.661e-06,
00706     6.9e-06, 7.146e-06, 7.276e-06, 7.374e-06, 7.447e-06, 7.383e-06,
00707     7.321e-06, 7.161e-06, 6.879e-06, 6.611e-06, 6.216e-06, 5.765e-06,
```

```
00708     5.355e-06, 4.905e-06, 4.471e-06, 4.075e-06, 3.728e-06, 3.413e-06,
00709     3.125e-06, 2.856e-06, 2.607e-06, 2.379e-06, 2.17e-06, 1.978e-06,
00710     1.8e-06, 1.646e-06, 1.506e-06, 1.376e-06, 1.233e-06, 1.102e-06,
00711     9.839e-07, 8.771e-07, 7.814e-07, 6.947e-07, 6.102e-07, 5.228e-07,
00712     4.509e-07, 3.922e-07, 3.501e-07, 3.183e-07, 2.909e-07, 2.686e-07,
00713     2.476e-07, 2.284e-07, 2.109e-07, 2.003e-07, 2.013e-07, 2.022e-07,
00714     2.032e-07, 2.042e-07, 2.097e-07, 2.361e-07, 2.656e-07, 2.989e-07,
00715     3.37e-07, 3.826e-07, 4.489e-07, 5.26e-07, 6.189e-07, 7.312e-07,
00716     8.496e-07, 8.444e-07, 8.392e-07, 8.339e-07, 8.286e-07, 8.234e-07,
00717     8.181e-07, 8.129e-07, 8.077e-07, 8.026e-07, 6.918e-07, 5.176e-07,
00718     3.865e-07, 2.885e-07, 2.156e-07, 1.619e-07, 1.219e-07, 9.161e-08,
00719     6.972e-08, 5.399e-08, 3.498e-08, 2.111e-08, 1.322e-08, 8.482e-09,
00720     5.527e-09, 3.423e-09, 2.071e-09, 1.314e-09, 8.529e-10, 5.503e-10,
00721     3.665e-10
00722 };
00723
00724 static double ocs[121] = {
00725     6e-10, 6e-10, 6e-10, 6e-10, 6e-10, 6e-10, 6e-10, 6e-10, 6e-10, 5.997e-10,
00726     5.989e-10, 5.881e-10, 5.765e-10, 5.433e-10, 5.074e-10, 4.567e-10,
00727     4.067e-10, 3.601e-10, 3.093e-10, 2.619e-10, 2.232e-10, 1.805e-10,
00728     1.46e-10, 1.187e-10, 8.03e-11, 5.435e-11, 3.686e-11, 2.217e-11,
00729     1.341e-11, 8.756e-12, 4.511e-12, 2.37e-12, 1.264e-12, 8.28e-13,
00730     5.263e-13, 3.209e-13, 1.717e-13, 9.068e-14, 4.709e-14, 2.389e-14,
00731     1.236e-14, 1.127e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00732     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00733     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00734     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00735     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00736     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00737     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00738     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00739     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00740     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00741     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00742     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00743     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00744     1.091e-14, 1.091e-14, 1.091e-14
00745 };
00746
00747 static double sf6[121] = {
00748     4.103e-12, 4.103e-12, 4.103e-12, 4.103e-12, 4.103e-12, 4.103e-12,
00749     4.103e-12, 4.103e-12, 4.103e-12, 4.087e-12, 4.064e-12, 4.023e-12,
00750     3.988e-12, 3.941e-12, 3.884e-12, 3.755e-12, 3.622e-12, 3.484e-12,
00751     3.32e-12, 3.144e-12, 2.978e-12, 2.811e-12, 2.653e-12, 2.489e-12,
00752     2.332e-12, 2.199e-12, 2.089e-12, 2.013e-12, 1.953e-12, 1.898e-12,
00753     1.859e-12, 1.826e-12, 1.798e-12, 1.776e-12, 1.757e-12, 1.742e-12,
00754     1.728e-12, 1.717e-12, 1.707e-12, 1.698e-12, 1.691e-12, 1.685e-12,
00755     1.679e-12, 1.675e-12, 1.671e-12, 1.668e-12, 1.665e-12, 1.663e-12,
00756     1.661e-12, 1.659e-12, 1.658e-12, 1.657e-12, 1.656e-12, 1.655e-12,
00757     1.654e-12, 1.653e-12, 1.653e-12, 1.652e-12, 1.652e-12, 1.652e-12,
00758     1.651e-12, 1.651e-12, 1.651e-12, 1.651e-12, 1.651e-12, 1.651e-12,
00759     1.651e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00760     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00761     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00762     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00763     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00764     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00765     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00766     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12
00767 };
00768
00769 static double so2[121] = {
00770     1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10,
00771     1e-10, 1e-10, 9.867e-11, 9.537e-11, 9e-11, 8.404e-11, 7.799e-11,
00772     7.205e-11, 6.616e-11, 6.036e-11, 5.475e-11, 5.007e-11, 4.638e-11,
00773     4.346e-11, 4.055e-11, 3.763e-11, 3.471e-11, 3.186e-11, 2.905e-11,
00774     2.631e-11, 2.358e-11, 2.415e-11, 2.949e-11, 3.952e-11, 5.155e-11,
00775     6.76e-11, 8.741e-11, 1.099e-10, 1.278e-10, 1.414e-10, 1.512e-10,
00776     1.607e-10, 1.699e-10, 1.774e-10, 1.832e-10, 1.871e-10, 1.907e-10,
00777     1.943e-10, 1.974e-10, 1.993e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00778     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00779     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00780     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00781     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00782     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00783     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00784     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10
00785 };
00786
00787 static int ig_co2 = -999;
00788
00789 double *q[NG] = { NULL };
00790
00791 /* Find emitter index of CO2... */
00792 if (ig_co2 == -999)
00793     ig_co2 = find_emitter(ct1, "CO2");
00794
```



```

00795  /* Identify variable... */
00796  for (int ig = 0; ig < ctl->ng; ig++) {
00797      q[ig] = NULL;
00798      if (strcasecmp(ctl->emitter[ig], "C2H2") == 0)
00799          q[ig] = c2h2;
00800      if (strcasecmp(ctl->emitter[ig], "C2H6") == 0)
00801          q[ig] = c2h6;
00802      if (strcasecmp(ctl->emitter[ig], "CCl4") == 0)
00803          q[ig] = ccl4;
00804      if (strcasecmp(ctl->emitter[ig], "CH4") == 0)
00805          q[ig] = ch4;
00806      if (strcasecmp(ctl->emitter[ig], "ClO") == 0)
00807          q[ig] = clo;
00808      if (strcasecmp(ctl->emitter[ig], "ClONO2") == 0)
00809          q[ig] = clono2;
00810      if (strcasecmp(ctl->emitter[ig], "CO") == 0)
00811          q[ig] = co;
00812      if (strcasecmp(ctl->emitter[ig], "COF2") == 0)
00813          q[ig] = cof2;
00814      if (strcasecmp(ctl->emitter[ig], "F11") == 0)
00815          q[ig] = f11;
00816      if (strcasecmp(ctl->emitter[ig], "F12") == 0)
00817          q[ig] = f12;
00818      if (strcasecmp(ctl->emitter[ig], "F14") == 0)
00819          q[ig] = f14;
00820      if (strcasecmp(ctl->emitter[ig], "F22") == 0)
00821          q[ig] = f22;
00822      if (strcasecmp(ctl->emitter[ig], "H2O") == 0)
00823          q[ig] = h2o;
00824      if (strcasecmp(ctl->emitter[ig], "H2O2") == 0)
00825          q[ig] = h2o2;
00826      if (strcasecmp(ctl->emitter[ig], "HCN") == 0)
00827          q[ig] = hcn;
00828      if (strcasecmp(ctl->emitter[ig], "HNO3") == 0)
00829          q[ig] = hno3;
00830      if (strcasecmp(ctl->emitter[ig], "HNO4") == 0)
00831          q[ig] = hno4;
00832      if (strcasecmp(ctl->emitter[ig], "HOCl") == 0)
00833          q[ig] = hocl;
00834      if (strcasecmp(ctl->emitter[ig], "N2O") == 0)
00835          q[ig] = n2o;
00836      if (strcasecmp(ctl->emitter[ig], "N2O5") == 0)
00837          q[ig] = n2o5;
00838      if (strcasecmp(ctl->emitter[ig], "NH3") == 0)
00839          q[ig] = nh3;
00840      if (strcasecmp(ctl->emitter[ig], "NO") == 0)
00841          q[ig] = no;
00842      if (strcasecmp(ctl->emitter[ig], "NO2") == 0)
00843          q[ig] = no2;
00844      if (strcasecmp(ctl->emitter[ig], "O3") == 0)
00845          q[ig] = o3;
00846      if (strcasecmp(ctl->emitter[ig], "OCS") == 0)
00847          q[ig] = ocs;
00848      if (strcasecmp(ctl->emitter[ig], "SF6") == 0)
00849          q[ig] = sf6;
00850      if (strcasecmp(ctl->emitter[ig], "SO2") == 0)
00851          q[ig] = so2;
00852  }
00853
00854  /* Loop over atmospheric data points... */
00855  for (int ip = 0; ip < atm->np; ip++) {
00856
00857      /* Get altitude index... */
00858      int iz = locate_reg(z, 121, atm->z[ip]);
00859
00860      /* Interpolate pressure... */
00861      atm->p[ip] = EXP(z[iz], pre[iz], z[iz + 1], pre[iz + 1], atm->z[ip]);
00862
00863      /* Interpolate temperature... */
00864      atm->t[ip] = LIN(z[iz], tem[iz], z[iz + 1], tem[iz + 1], atm->z[ip]);
00865
00866      /* Interpolate trace gases... */
00867      for (int ig = 0; ig < ctl->ng; ig++)
00868          if (q[ig] != NULL)
00869              atm->q[ig][ip] =
00870                  LIN(z[iz], q[ig][iz], z[iz + 1], q[ig][iz + 1], atm->z[ip]);
00871          else
00872              atm->q[ig][ip] = 0;
00873
00874      /* Set CO2... */
00875      if (ig_co2 >= 0)
00876          atm->q[ig_co2][ip] =
00877              371.789948e-6 + 2.026214e-6 * (atm->time[ip] - 63158400.) / 31557600.;
00878
00879      /* Set extinction to zero... */
00880      for (int iw = 0; iw < ctl->nw; iw++)
00881          atm->k[iw][ip] = 0;

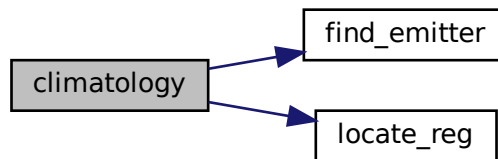
```

```

00882
00883     /* Set cloud layer... */
00884     atm->clz = atm->cldz = 0;
00885     for (int icl = 0; icl < ctl->ncl; icl++)
00886         atm->clk[icl] = 0;
00887
00888     /* Set surface layer... */
00889     atm->sfz = atm->sfp = atm->sft = 0;
00890     for (int isf = 0; isf < ctl->nsf; isf++)
00891         atm->sfeps[isf] = 1;
00892 }
00893 }

```

Here is the call graph for this function:



**5.17.2.6 ctmc02()** double ctmc02 (

```

    double nu,
    double p,
    double t,
    double u )

```

Compute carbon dioxide continuum (optical depth).

Definition at line 897 of file [jurassic.c](#).

```

00901     {
00902
00903     static double co2296[2001] = { 9.3388e-5, 9.7711e-5, 1.0224e-4, 1.0697e-4,
00904     1.1193e-4, 1.1712e-4, 1.2255e-4, 1.2824e-4, 1.3419e-4, 1.4043e-4,
00905     1.4695e-4, 1.5378e-4, 1.6094e-4, 1.6842e-4, 1.7626e-4, 1.8447e-4,
00906     1.9307e-4, 2.0207e-4, 2.1149e-4, 2.2136e-4, 2.3169e-4, 2.4251e-4,
00907     2.5384e-4, 2.657e-4, 2.7813e-4, 2.9114e-4, 3.0477e-4, 3.1904e-4,
00908     3.3399e-4, 3.4965e-4, 3.6604e-4, 3.8322e-4, 4.0121e-4, 4.2006e-4,
00909     4.398e-4, 4.6047e-4, 4.8214e-4, 5.0483e-4, 5.286e-4, 5.535e-4,
00910     5.7959e-4, 6.0693e-4, 6.3557e-4, 6.6558e-4, 6.9702e-4, 7.2996e-4,
00911     7.6449e-4, 8.0066e-4, 8.3856e-4, 8.7829e-4, 9.1991e-4, 9.6354e-4,
00912     .0010093, .0010572, .0011074, .00116, .0012152, .001273,
00913     .0013336, .0013972, .0014638, .0015336, .0016068, .0016835,
00914     .001764, .0018483, .0019367, .0020295, .0021267, .0022286,
00915     .0023355, .0024476, .0025652, .0026885, .0028178, .0029534,
00916     .0030956, .0032448, .0034012, .0035654, .0037375, .0039181,
00917     .0041076, .0043063, .0045148, .0047336, .0049632, .005204,
00918     .0054567, .0057219, .0060002, .0062923, .0065988, .0069204,
00919     .007258, .0076123, .0079842, .0083746, .0087844, .0092146,
00920     .0096663, .01014, .010638, .011161, .01171, .012286, .012891,
00921     .013527, .014194, .014895, .015631, .016404, .017217, .01807,
00922     .018966, .019908, .020897, .021936, .023028, .024176, .025382,
00923     .026649, .027981, .02938, .030851, .032397, .034023, .035732,
00924     .037528, .039416, .041402, .04349, .045685, .047994, .050422,
00925     .052975, .055661, .058486, .061458, .064584, .067873, .071334,
00926     .074975, .078807, .082839, .087082, .091549, .096249, .1012,
00927     .10641, .11189, .11767, .12375, .13015, .13689, .14399, .15147,
00928     .15935, .16765, .17639, .18561, .19531, .20554, .21632, .22769,
00929     .23967, .25229, .2656, .27964, .29443, .31004, .3265, .34386,
00930     .36218, .3815, .40188, .42339, .44609, .47004, .49533, .52202,
00931     .5502, .57995, .61137, .64455, .6796, .71663, .75574, .79707,

```

00932 .84075, .88691, .9357, .98728, 1.0418, 1.0995, 1.1605, 1.225,  
00933 1.2932, 1.3654, 1.4418, 1.5227, 1.6083, 1.6989, 1.7948, 1.8964,  
00934 2.004, 2.118, 2.2388, 2.3668, 2.5025, 2.6463, 2.7988, 2.9606,  
00935 3.1321, 3.314, 3.5071, 3.712, 3.9296, 4.1605, 4.4058, 4.6663,  
00936 4.9431, 5.2374, 5.5501, 5.8818, 6.2353, 6.6114, 7.0115, 7.4372,  
00937 7.8905, 8.3731, 8.8871, 9.4349, 10.019, 10.641, 11.305, 12.013,  
00938 12.769, 13.576, 14.437, 15.358, 16.342, 17.39, 18.513, 19.716,  
00939 21.003, 22.379, 23.854, 25.436, 27.126, 28.942, 30.89, 32.973,  
00940 35.219, 37.634, 40.224, 43.021, 46.037, 49.29, 52.803, 56.447,  
00941 60.418, 64.792, 69.526, 74.637, 80.182, 86.193, 92.713, 99.786,  
00942 107.47, 115.84, 124.94, 134.86, 145.69, 157.49, 170.3, 184.39,  
00943 199.83, 216.4, 234.55, 254.72, 276.82, 299.85, 326.16, 354.99,  
00944 386.51, 416.68, 449.89, 490.12, 534.35, 578.25, 632.26, 692.61,  
00945 756.43, 834.75, 924.11, 1016.9, 996.96, 1102.7, 1219.2, 1351.9,  
00946 1494.3, 1654.1, 1826.5, 2027.9, 2249., 2453.8, 2714.4, 2999.4,  
00947 3209.5, 3509., 3840.4, 3907.5, 4190.7, 4533.5, 4648.3, 5059.1,  
00948 5561.6, 6191.4, 6820.8, 7905.9, 9362.2, 2431.3, 2211.3, 2046.8,  
00949 2023.8, 1985.9, 1905.9, 1491.1, 1369.8, 1262.2, 1200.7, 887.74,  
00950 820.25, 885.23, 887.21, 816.73, 1126.9, 1216.2, 1272.4, 1579.5,  
00951 1634.2, 1656.3, 1657.9, 1789.5, 1670.8, 1509.5, 8474.6, 7489.2,  
00952 6793.6, 6117., 5574.1, 5141.2, 5084.6, 4745.1, 4413.2, 4102.8,  
00953 4024.7, 3715., 3398.6, 3100.8, 2900.4, 2629.2, 2374., 2144.7,  
00954 1955.8, 1760.8, 1591.2, 1435.2, 1296.2, 1174., 1065.1, 967.76,  
00955 999.48, 897.45, 809.23, 732.77, 670.26, 611.93, 560.11, 518.77,  
00956 476.84, 438.8, 408.48, 380.21, 349.24, 322.71, 296.65, 272.85,  
00957 251.96, 232.04, 213.88, 197.69, 182.41, 168.41, 155.79, 144.05,  
00958 133.31, 123.48, 114.5, 106.21, 98.591, 91.612, 85.156, 79.204,  
00959 73.719, 68.666, 63.975, 59.637, 56.35, 52.545, 49.042, 45.788,  
00960 42.78, 39.992, 37.441, 35.037, 32.8, 30.744, 28.801, 26.986,  
00961 25.297, 23.731, 22.258, 20.883, 19.603, 18.403, 17.295, 16.249,  
00962 15.271, 14.356, 13.501, 12.701, 11.954, 11.254, 10.6, 9.9864,  
00963 9.4118, 8.8745, 8.3714, 7.8997, 7.4578, 7.0446, 6.6573, 6.2949,  
00964 5.9577, 5.6395, 5.3419, 5.063, 4.8037, 4.5608, 4.3452, 4.1364,  
00965 3.9413, 3.7394, 3.562, 3.3932, 3.2325, 3.0789, 2.9318, 2.7898,  
00966 2.6537, 2.5225, 2.3958, 2.2305, 2.1215, 2.0245, 1.9427, 1.8795,  
00967 1.8336, 1.7604, 1.7016, 1.6419, 1.5282, 1.4611, 1.3443, 1.27,  
00968 1.1675, 1.0824, 1.0534, .99833, .95854, .92981, .90887, .89346,  
00969 .88113, .87068, .86102, .85096, .88262, .86151, .83565, .80518,  
00970 .77045, .73736, .74744, .74954, .75773, .82267, .83493, .89402,  
00971 .89725, .93426, .95564, .94045, .94174, .93404, .92035, .90456,  
00972 .88621, .86673, .78117, .7515, .72056, .68822, .65658, .62764,  
00973 .55984, .55598, .57407, .60963, .63763, .66198, .61132, .60972,  
00974 .52496, .50649, .41872, .3964, .32422, .27276, .24048, .23772,  
00975 .2286, .22711, .23999, .32038, .34371, .36621, .38561, .39953,  
00976 .40636, .44913, .42716, .3919, .35477, .33935, .3351, .39746,  
00977 .40993, .49398, .49956, .56157, .54742, .57295, .57386, .55417,  
00978 .50745, .471, .43446, .39102, .34993, .31269, .27888, .24912,  
00979 .22291, .19994, .17972, .16197, .14633, .13252, .12029, .10942,  
00980 .099745, .091118, .083404, .076494, .070292, .064716, .059697,  
00981 .055173, .051093, .047411, .044089, .041092, .038392, .035965,  
00982 .033789, .031846, .030122, .028607, .02729, .026169, .025209,  
00983 .024405, .023766, .023288, .022925, .022716, .022681, .022685,  
00984 .022768, .023133, .023325, .023486, .024004, .024126, .024083,  
00985 .023785, .024023, .023029, .021649, .021108, .019454, .017809,  
00986 .017292, .016635, .017037, .018068, .018977, .018756, .017847,  
00987 .016557, .016142, .014459, .012869, .012381, .010875, .0098701,  
00988 .009285, .0091698, .0091701, .0096145, .010553, .01106, .012613,  
00989 .014362, .015017, .016507, .017741, .01768, .017784, .0171,  
00990 .016357, .016172, .017257, .018978, .020935, .021741, .023567,  
00991 .025183, .025589, .026732, .027648, .028278, .028215, .02856,  
00992 .029015, .029062, .028851, .028497, .027825, .027801, .026523,  
00993 .02487, .022967, .022168, .020194, .018605, .017903, .018439,  
00994 .019697, .020311, .020855, .020057, .018608, .016738, .015963,  
00995 .013844, .011801, .011134, .0097573, .0086007, .0086226,  
00996 .0083721, .0090978, .0097616, .0098426, .011317, .012853, .01447,  
00997 .014657, .015771, .016351, .016079, .014829, .013431, .013185,  
00998 .013207, .01448, .016176, .017971, .018265, .019526, .020455,  
00999 .019797, .019802, .0194, .018176, .017505, .016197, .015339,  
01000 .014401, .013213, .012203, .011186, .010236, .0093288, .0084854,  
01001 .0076837, .0069375, .0062614, .0056628, .0051153, .0046015,  
01002 .0041501, .003752, .0033996, .0030865, .0028077, .0025586,  
01003 .0023355, .0021353, .0019553, .0017931, .0016466, .0015141,  
01004 .0013941, .0012852, .0011862, .0010962, .0010142, 9.3935e-4,  
01005 8.71e-4, 8.0851e-4, 7.5132e-4, 6.9894e-4, 6.5093e-4, 6.0689e-4,  
01006 5.6647e-4, 5.2935e-4, 4.9525e-4, 4.6391e-4, 4.3509e-4, 4.086e-4,  
01007 3.8424e-4, 3.6185e-4, 3.4126e-4, 3.2235e-4, 3.0498e-4, 2.8904e-4,  
01008 2.7444e-4, 2.6106e-4, 2.4883e-4, 2.3766e-4, 2.275e-4, 2.1827e-4,  
01009 2.0992e-4, 2.0239e-4, 1.9563e-4, 1.896e-4, 1.8427e-4, 1.796e-4,  
01010 1.7555e-4, 1.7209e-4, 1.692e-4, 1.6687e-4, 1.6505e-4, 1.6375e-4,  
01011 1.6294e-4, 1.6261e-4, 1.6274e-4, 1.6334e-4, 1.6438e-4, 1.6587e-4,  
01012 1.678e-4, 1.7017e-4, 1.7297e-4, 1.762e-4, 1.7988e-4, 1.8399e-4,  
01013 1.8855e-4, 1.9355e-4, 1.9902e-4, 2.0494e-4, 2.1134e-4, 2.1823e-4,  
01014 2.2561e-4, 2.335e-4, 2.4192e-4, 2.5088e-4, 2.604e-4, 2.705e-4,  
01015 2.8119e-4, 2.9251e-4, 3.0447e-4, 3.171e-4, 3.3042e-4, 3.4447e-4,  
01016 3.5927e-4, 3.7486e-4, 3.9127e-4, 4.0854e-4, 4.267e-4, 4.4579e-4,  
01017 4.6586e-4, 4.8696e-4, 5.0912e-4, 5.324e-4, 5.5685e-4, 5.8253e-4,  
01018 6.0949e-4, 6.378e-4, 6.6753e-4, 6.9873e-4, 7.3149e-4, 7.6588e-4,

```
01019      8.0198e-4, 8.3987e-4, 8.7964e-4, 9.2139e-4, 9.6522e-4, .0010112,
01020      .0010595, .0011102, .0011634, .0012193, .001278, .0013396,
01021      .0014043, .0014722, .0015436, .0016185, .0016972, .0017799,
01022      .0018668, .001958, .0020539, .0021547, .0022606, .0023719,
01023      .002489, .002612, .0027414, .0028775, .0030206, .0031712,
01024      .0033295, .0034962, .0036716, .0038563, .0040506, .0042553,
01025      .0044709, .004698, .0049373, .0051894, .0054552, .0057354,
01026      .006031, .0063427, .0066717, .0070188, .0073854, .0077726,
01027      .0081816, .0086138, .0090709, .0095543, .010066, .010607,
01028      .011181, .011789, .012433, .013116, .013842, .014613, .015432,
01029      .016304, .017233, .018224, .019281, .020394, .021574, .022836,
01030      .024181, .025594, .027088, .028707, .030401, .032245, .034219,
01031      .036262, .038539, .040987, .043578, .04641, .04949, .052726,
01032      .056326, .0602, .064093, .068521, .073278, .077734, .083064,
01033      .088731, .093885, .1003, .1072, .11365, .12187, .13078, .13989,
01034      .15095, .16299, .17634, .19116, .20628, .22419, .24386, .26587,
01035      .28811, .31399, .34321, .36606, .39675, .42742, .44243, .47197,
01036      .49993, .49027, .51147, .52803, .48931, .49729, .5026, .43854,
01037      .441, .44766, .43414, .46151, .50029, .55247, .43855, .32115,
01038      .32607, .3431, .36119, .38029, .41179, .43996, .47144, .51853,
01039      .55362, .59122, .66338, .69877, .74001, .82923, .86907, .90361,
01040      1.0025, 1.031, 1.0559, 1.104, 1.1178, 1.1341, 1.1547, 1.351,
01041      1.4772, 1.4812, 1.4907, 1.512, 1.5442, 1.5853, 1.6358, 1.6963,
01042      1.7674, 1.8474, 1.9353, 2.0335, 2.143, 2.2592, 2.3853, 2.5217,
01043      2.6686, 2.8273, 2.9998, 3.183, 3.3868, 3.6109, 3.8564, 4.1159,
01044      4.4079, 4.7278, 5.0497, 5.3695, 5.758, 6.0834, 6.4976, 6.9312,
01045      7.38, 7.5746, 7.9833, 8.3791, 8.3956, 8.7501, 9.1067, 9.072,
01046      9.4649, 9.9112, 10.402, 10.829, 11.605, 12.54, 12.713, 10.443,
01047      10.825, 11.375, 11.955, 12.623, 13.326, 14.101, 15.041, 15.547,
01048      16.461, 17.439, 18.716, 19.84, 21.036, 22.642, 23.901, 25.244,
01049      27.03, 28.411, 29.871, 31.403, 33.147, 34.744, 36.456, 39.239,
01050      43.605, 45.162, 47.004, 49.093, 51.391, 53.946, 56.673, 59.629,
01051      63.167, 66.576, 70.254, 74.222, 78.477, 83.034, 87.914, 93.18,
01052      98.77, 104.74, 111.15, 117.95, 125.23, 133.01, 141.33, 150.21,
01053      159.71, 169.89, 180.93, 192.54, 204.99, 218.34, 232.65, 248.,
01054      264.47, 282.14, 301.13, 321.53, 343.48, 367.08, 392.5, 419.88,
01055      449.4, 481.26, 515.64, 552.79, 592.99, 636.48, 683.61, 734.65,
01056      789.99, 850.02, 915.14, 985.81, 1062.5, 1147.1, 1237.8, 1336.4,
01057      1443.2, 1558.9, 1684.2, 1819.2, 1965.2, 2122.6, 2291.7, 2470.8,
01058      2665.7, 2874.9, 3099.4, 3337.9, 3541., 3813.3, 4111.9, 4439.3,
01059      4798.9, 5196., 5639.2, 6087.5, 6657.7, 7306.7, 8040.7, 8845.5,
01060      9702.2, 10670., 11739., 12842., 14141., 15498., 17068., 18729.,
01061      20557., 22559., 25248., 27664., 30207., 32915., 35611., 38081.,
01062      40715., 43191., 41651., 42750., 43785., 44353., 44366., 44189.,
01063      43618., 42862., 41878., 35133., 35215., 36383., 39420., 44055.,
01064      44155., 45850., 46853., 39197., 38274., 29942., 28553., 21792.,
01065      21228., 17106., 14955., 18181., 19557., 21427., 23728., 26301.,
01066      28584., 30775., 32536., 33867., 40089., 39204., 37329., 34452.,
01067      31373., 33921., 34800., 36043., 44415., 45162., 52181., 50895.,
01068      54140., 50840., 50468., 48302., 44915., 40910., 36754., 32755.,
01069      29093., 25860., 22962., 20448., 18247., 16326., 14645., 13165.,
01070      11861., 10708., 9686.9, 8779.7, 7971.9, 7250.8, 6605.7, 6027.2,
01071      5507.3, 5039.1, 4616.6, 4234.8, 3889., 3575.4, 3290.5, 3031.3,
01072      2795.2, 2579.9, 2383.1, 2203.3, 2038.6, 1887.6, 1749.1, 1621.9,
01073      1505., 1397.4, 1298.3, 1207., 1122.8, 1045., 973.1, 906.64,
01074      845.16, 788.22, 735.48, 686.57, 641.21, 599.1, 559.99, 523.64,
01075      489.85, 458.42, 429.16, 401.92, 376.54, 352.88, 330.82, 310.24,
01076      291.03, 273.09, 256.34, 240.69, 226.05, 212.37, 199.57, 187.59,
01077      176.37, 165.87, 156.03, 146.82, 138.17, 130.07, 122.47, 115.34,
01078      108.65, 102.37, 96.473, 90.934, 85.73, 80.84, 76.243, 71.922,
01079      67.858, 64.034, 60.438, 57.052, 53.866, 50.866, 48.04, 45.379,
01080      42.872, 40.51, 38.285, 36.188, 34.211, 32.347, 30.588, 28.929,
01081      27.362, 25.884, 24.489, 23.171, 21.929, 20.755, 19.646, 18.599,
01082      17.61, 16.677, 15.795, 14.961, 14.174, 13.43, 12.725, 12.06,
01083      11.431, 10.834, 10.27, 9.7361, 9.2302, 8.7518, 8.2997, 7.8724,
01084      7.4674, 7.0848, 6.7226, 6.3794, 6.054, 5.745, 5.4525, 5.1752,
01085      4.9121, 4.6625, 4.4259, 4.2015, 3.9888, 3.7872, 3.5961, 3.4149,
01086      3.2431, 3.0802, 2.9257, 2.7792, 2.6402, 2.5084, 2.3834, 2.2648,
01087      2.1522, 2.0455, 1.9441, 1.848, 1.7567, 1.6701, 1.5878, 1.5097,
01088      1.4356, 1.3651, 1.2981, 1.2345, 1.174, 1.1167, 1.062, 1.0101,
01089      .96087, .91414, .86986, .82781, .78777, .74971, .71339, .67882,
01090      .64604, .61473, .58507, .55676, .52987, .5044, .48014, .45715,
01091      .43527, .41453, .3948, .37609, .35831, .34142, .32524, .30995,
01092      .29536, .28142, .26807, .25527, .24311, .23166, .22077, .21053,
01093      .20081, .19143, .18261, .17407, .16603, .15833, .15089, .14385,
01094      .13707, .13065, .12449, .11865, .11306, .10774, .10266, .097818,
01095      .093203, .088815, .084641, .080671, .076892, .073296, .069873,
01096      .066613, .06351, .060555, .05774, .055058, .052504, .050071,
01097      .047752, .045543, .043438, .041432, .039521, .037699, .035962,
01098      .034307, .032729, .031225, .029791, .028423, .02712, .025877,
01099      .024692, .023563, .022485, .021458, .020478, .019543, .018652,
01100      .017802, .016992, .016219, .015481, .014778, .014107, .013467,
01101      .012856, .012274, .011718, .011188, .010682, .0102, .0097393,
01102      .0093001, .008881, .0084812, .0080997, .0077358, .0073885,
01103      .0070571, .0067409, .0064393, .0061514, .0058768, .0056147,
01104      .0053647, .0051262, .0048987, .0046816, .0044745, .0042769,
01105      .0040884, .0039088, .0037373, .0035739, .003418, .0032693,
```

```

01106 .0031277, .0029926, .0028639, .0027413, .0026245, .0025133,
01107 .0024074, .0023066, .0022108, .0021196, .002033, .0019507,
01108 .0018726, .0017985, .0017282, .0016617, .0015988, .0015394,
01109 .0014834, .0014306, .0013811, .0013346, .0012911, .0012506,
01110 .0012131, .0011784, .0011465, .0011175, .0010912, .0010678,
01111 .0010472, .0010295, .0010147, .001003, 9.9428e-4, 9.8883e-4,
01112 9.8673e-4, 9.8821e-4, 9.9343e-4, .0010027, .0010164, .0010348,
01113 .0010586, .0010882, .0011245, .0011685, .0012145, .0012666,
01114 .0013095, .0013688, .0014048, .0014663, .0015309, .0015499,
01115 .0016144, .0016312, .001705, .0017892, .0018499, .0019715,
01116 .0021102, .0022442, .0024284, .0025893, .0027703, .0029445,
01117 .0031193, .003346, .0034552, .0036906, .0037584, .0040084,
01118 .0041934, .0044587, .0047093, .0049759, .0053421, .0055134,
01119 .0059048, .0058663, .0061036, .0063259, .0059657, .0060653,
01120 .0060972, .0055539, .0055653, .0055772, .005331, .0054953,
01121 .0055919, .0058684, .006183, .0066675, .0069808, .0075142,
01122 .0078536, .0084282, .0089454, .0094625, .0093703, .0095857,
01123 .0099283, .010063, .010521, .0097778, .0098175, .010379, .010447,
01124 .0105, .010617, .010706, .01078, .011177, .011212, .011304,
01125 .011446, .011603, .011816, .012165, .012545, .013069, .013539,
01126 .01411, .014776, .016103, .017016, .017994, .018978, .01998,
01127 .021799, .022745, .023681, .024627, .025562, .026992, .027958,
01128 .029013, .030154, .031402, .03228, .033651, .035272, .037088,
01129 .039021, .041213, .043597, .045977, .04877, .051809, .054943,
01130 .058064, .061528, .06537, .069309, .071928, .075752, .079589,
01131 .083352, .084096, .087497, .090817, .091198, .094966, .099045,
01132 .10429, .10867, .11518, .12269, .13126, .14087, .15161, .16388,
01133 .16423, .1759, .18721, .19994, .21275, .22513, .23041, .24231,
01134 .25299, .25396, .26396, .27696, .27929, .2908, .30595, .31433,
01135 .3282, .3429, .35944, .37467, .39277, .41245, .43326, .45649,
01136 .48152, .51897, .54686, .57877, .61263, .64962, .68983, .73945,
01137 .78619, .83537, .89622, .95002, 1.0067, 1.0742, 1.1355, 1.2007,
01138 1.2738, 1.347, 1.4254, 1.5094, 1.6009, 1.6976, 1.8019, 1.9148,
01139 2.0357, 2.166, 2.3066, 2.4579, 2.6208, 2.7966, 2.986, 3.188,
01140 3.4081, 3.6456, 3.9, 4.1747, 4.4712, 4.7931, 5.1359, 5.5097,
01141 5.9117, 6.3435, 6.8003, 7.3001, 7.8385, 8.3945, 9.011, 9.6869,
01142 10.392, 11.18, 12.036, 12.938, 13.944, 14.881, 16.029, 17.255,
01143 18.574, 19.945, 21.38, 22.9, 24.477, 26.128, 27.87, 29.037,
01144 30.988, 33.145, 35.506, 37.76, 40.885, 44.487, 48.505, 52.911,
01145 57.56, 61.964, 67.217, 72.26, 78.343, 85.08, 91.867, 99.435,
01146 107.68, 116.97, 127.12, 138.32, 150.26, 163.04, 174.81, 189.26,
01147 205.61, 224.68, 240.98, 261.88, 285.1, 307.58, 334.35, 363.53,
01148 394.68, 427.85, 458.85, 489.25, 472.87, 486.93, 496.27, 501.52,
01149 501.57, 497.14, 488.09, 476.32, 393.76, 388.51, 393.42, 414.45,
01150 455.12, 514.62, 520.38, 547.42, 562.6, 487.47, 480.83, 391.06,
01151 376.92, 303.7, 295.91, 256.03, 236.73, 280.38, 310.71, 335.53,
01152 367.88, 401.94, 435.52, 469.13, 497.94, 588.82, 597.94, 597.2,
01153 588.28, 571.2, 555.75, 603.56, 638.15, 680.75, 801.72, 848.01,
01154 962.15, 990.06, 1068.1, 1076.2, 1115.3, 1134.2, 1136.6, 1119.1,
01155 1108.9, 1090.6, 1068.7, 1041.9, 1005.4, 967.98, 927.08, 780.1,
01156 751.41, 733.12, 742.65, 785.56, 855.16, 852.45, 878.1, 784.59,
01157 777.81, 765.13, 622.93, 498.09, 474.89, 386.9, 378.48, 336.17,
01158 322.04, 329.57, 350.5, 383.38, 420.02, 462.39, 499.71, 531.98,
01159 654.99, 653.43, 639.99, 605.16, 554.16, 504.42, 540.64, 552.33,
01160 679.46, 699.51, 713.91, 832.17, 919.91, 884.96, 907.57, 846.56,
01161 818.56, 768.93, 706.71, 642.17, 575.95, 515.38, 459.07, 409.02,
01162 364.61, 325.46, 291.1, 260.89, 234.39, 211.01, 190.38, 172.11,
01163 155.91, 141.49, 128.63, 117.13, 106.84, 97.584, 89.262, 81.756,
01164 74.975, 68.842, 63.28, 58.232, 53.641, 49.46, 45.649, 42.168,
01165 38.991, 36.078, 33.409, 30.96, 28.71, 26.642, 24.737, 22.985,
01166 21.37, 19.882, 18.512, 17.242, 16.073, 14.987, 13.984, 13.05,
01167 12.186, 11.384, 10.637, 9.9436, 9.2988, 8.6991, 8.141, 7.6215,
01168 7.1378, 6.6872, 6.2671, 5.8754, 5.51, 5.1691, 4.851, 4.5539,
01169 4.2764, 4.0169, 3.7742, 3.5472, 3.3348, 3.1359, 2.9495, 2.7749,
01170 2.6113, 2.4578, 2.3139, 2.1789, 2.0523, 1.9334, 1.8219, 1.7171,
01171 1.6188, 1.5263, 1.4395, 1.3579, 1.2812, 1.209, 1.1411, 1.0773,
01172 1.0171, .96048, .90713, .85684, .80959, .76495, .72282, .68309,
01173 .64563, .61035, .57707, .54573, .51622, .48834, .46199, .43709,
01174 .41359, .39129, .37034, .35064, .33198, .31442, .29784, .28218,
01175 .26732, .25337, .24017, .22774, .21601, .20479, .19426
01176 };
01177
01178 static double co2260[2001] = { 5.7971e-5, 6.0733e-5, 6.3628e-5, 6.6662e-5,
01179 6.9843e-5, 7.3176e-5, 7.6671e-5, 8.0334e-5, 8.4175e-5, 8.8201e-5,
01180 9.2421e-5, 9.6846e-5, 1.0149e-4, 1.0635e-4, 1.1145e-4, 1.1679e-4,
01181 1.224e-4, 1.2828e-4, 1.3444e-4, 1.409e-4, 1.4768e-4, 1.5479e-4,
01182 1.6224e-4, 1.7006e-4, 1.7826e-4, 1.8685e-4, 1.9587e-4, 2.0532e-4,
01183 2.1524e-4, 2.2565e-4, 2.3656e-4, 2.48e-4, 2.6001e-4, 2.7261e-4,
01184 2.8582e-4, 2.9968e-4, 3.1422e-4, 3.2948e-4, 3.4548e-4, 3.6228e-4,
01185 3.799e-4, 3.9838e-4, 4.1778e-4, 4.3814e-4, 4.595e-4, 4.8191e-4,
01186 5.0543e-4, 5.3012e-4, 5.5603e-4, 5.8321e-4, 6.1175e-4, 6.417e-4,
01187 6.7314e-4, 7.0614e-4, 7.4078e-4, 7.7714e-4, 8.1531e-4, 8.5538e-4,
01188 8.9745e-4, 9.4162e-4, 9.8798e-4, .0010367, .0010878, .0011415,
01189 .0011978, .001257, .0013191, .0013844, .001453, .0015249,
01190 .0016006, .00168, .0017634, .001851, .001943, .0020397, .0021412,
01191 .0022479, .00236, .0024778, .0026015, .0027316, .0028682,
01192 .0030117, .0031626, .0033211, .0034877, .0036628, .0038469,

```

```

01193 .0040403, .0042436, .0044574, .004682, .0049182, .0051665,
01194 .0054276, .0057021, .0059907, .0062942, .0066133, .0069489,
01195 .0073018, .0076729, .0080632, .0084738, .0089056, .0093599,
01196 .0098377, .01034, .010869, .011426, .012011, .012627, .013276,
01197 .013958, .014676, .015431, .016226, .017063, .017944, .018872,
01198 .019848, .020876, .021958, .023098, .024298, .025561, .026892,
01199 .028293, .029769, .031323, .032961, .034686, .036503, .038418,
01200 .040435, .042561, .044801, .047161, .049649, .052271, .055035,
01201 .057948, .061019, .064256, .06767, .07127, .075066, .079069,
01202 .083291, .087744, .092441, .097396, .10262, .10814, .11396,
01203 .1201, .12658, .13342, .14064, .14826, .1563, .1648, .17376,
01204 .18323, .19324, .2038, .21496, .22674, .23919, .25234, .26624,
01205 .28093, .29646, .31287, .33021, .34855, .36794, .38844, .41012,
01206 .43305, .45731, .48297, .51011, .53884, .56924, .60141, .63547,
01207 .67152, .70969, .75012, .79292, .83826, .8863, .93718, .99111,
01208 1.0482, 1.1088, 1.173, 1.2411, 1.3133, 1.3898, 1.471, 1.5571,
01209 1.6485, 1.7455, 1.8485, 1.9577, 2.0737, 2.197, 2.3278, 2.4668,
01210 2.6145, 2.7715, 2.9383, 3.1156, 3.3042, 3.5047, 3.7181, 3.9451,
01211 4.1866, 4.4437, 4.7174, 5.0089, 5.3192, 5.65, 6.0025, 6.3782,
01212 6.7787, 7.206, 7.6617, 8.1479, 8.6669, 9.221, 9.8128, 10.445,
01213 11.12, 11.843, 12.615, 13.441, 14.325, 15.271, 16.283, 17.367,
01214 18.529, 19.776, 21.111, 22.544, 24.082, 25.731, 27.504, 29.409,
01215 31.452, 33.654, 36.024, 38.573, 41.323, 44.29, 47.492, 50.951,
01216 54.608, 58.588, 62.929, 67.629, 72.712, 78.226, 84.207, 90.699,
01217 97.749, 105.42, 113.77, 122.86, 132.78, 143.61, 155.44, 168.33,
01218 182.48, 198.01, 214.87, 233.39, 253.86, 276.34, 300.3, 327.28,
01219 356.89, 389.48, 422.29, 458.99, 501.39, 548.13, 595.62, 652.74,
01220 716.54, 784.57, 866.78, 960.59, 1062.8, 1072.5, 1189.5, 1319.4,
01221 1467.6, 1630.2, 1813.7, 2016.9, 2253., 2515.3, 2773.5, 3092.8,
01222 3444.4, 3720.4, 4104.3, 4527.5, 4645.9, 5021.7, 5462.2, 5597.,
01223 6110.6, 6732.5, 7513.8, 8270.6, 9640.6, 11487., 2796.1, 2680.1,
01224 2441.6, 2404.2, 2334.8, 2215.2, 1642.5, 1477.9, 1328.1, 1223.5,
01225 843.34, 766.96, 831.65, 834.84, 774.85, 1156.3, 1275.6, 1366.1,
01226 1795.6, 1885., 1936.5, 1953.4, 2154.4, 2002.7, 1789.8, 10381,
01227 9040., 8216.5, 7384.7, 6721.9, 6187.7, 6143.8, 5703.9, 5276.6,
01228 4873.1, 4736., 4325.3, 3927., 3554.1, 3286.1, 2950.1, 2642.4,
01229 2368.7, 2138.9, 1914., 1719.6, 1543.9, 1388.6, 1252.1, 1132.2,
01230 1024.1, 1025.4, 920.58, 829.59, 750.54, 685.01, 624.25, 570.14,
01231 525.81, 481.85, 441.95, 408.71, 377.23, 345.86, 318.51, 292.26,
01232 268.34, 247.04, 227.14, 209.02, 192.69, 177.59, 163.78, 151.26,
01233 139.73, 129.19, 119.53, 110.7, 102.57, 95.109, 88.264, 81.948,
01234 76.13, 70.768, 65.827, 61.251, 57.022, 53.495, 49.824, 46.443,
01235 43.307, 40.405, 37.716, 35.241, 32.923, 30.77, 28.78, 26.915,
01236 25.177, 23.56, 22.059, 20.654, 19.345, 18.126, 16.988, 15.93,
01237 14.939, 14.014, 13.149, 12.343, 11.589, 10.884, 10.225, 9.6093,
01238 9.0327, 8.4934, 7.9889, 7.5166, 7.0744, 6.6604, 6.2727, 5.9098,
01239 5.5701, 5.2529, 4.955, 4.676, 4.4148, 4.171, 3.9426, 3.7332,
01240 3.5347, 3.3493, 3.1677, 3.0025, 2.8466, 2.6994, 2.5601, 2.4277,
01241 2.3016, 2.1814, 2.0664, 1.9564, 1.8279, 1.7311, 1.6427, 1.5645,
01242 1.4982, 1.443, 1.374, 1.3146, 1.2562, 1.17, 1.1105, 1.0272,
01243 .96863, .89718, .83654, .80226, .75908, .72431, .69573, .67174,
01244 .65126, .63315, .61693, .60182, .58715, .59554, .57649, .55526,
01245 .53177, .50622, .48176, .4813, .47642, .47492, .50273, .50293,
01246 .52687, .52239, .53419, .53814, .52626, .52211, .51492, .50622,
01247 .49746, .48841, .4792, .43534, .41999, .40349, .38586, .36799,
01248 .35108, .31089, .30803, .3171, .33599, .35041, .36149, .32924,
01249 .32462, .27309, .25961, .20922, .19504, .15683, .13098, .11588,
01250 .11478, .11204, .11363, .12135, .16423, .17785, .19094, .20236,
01251 .21084, .2154, .24108, .22848, .20871, .18797, .17963, .17834,
01252 .21552, .22284, .26945, .27052, .30108, .28977, .29772, .29224,
01253 .27658, .24956, .22777, .20654, .18392, .16338, .1452, .12916,
01254 .1152, .10304, .092437, .083163, .075031, .067878, .061564,
01255 .055976, .051018, .046609, .042679, .03917, .036032, .033223,
01256 .030706, .02845, .026428, .024617, .022998, .021554, .02027,
01257 .019136, .018114, .017278, .016541, .015926, .015432, .015058,
01258 .014807, .014666, .014635, .014728, .014947, .01527, .015728,
01259 .016345, .017026, .017798, .018839, .019752, .020636, .021886,
01260 .022695, .02327, .023478, .024292, .023544, .022222, .021932,
01261 .020052, .018143, .017722, .017031, .017782, .01938, .020734,
01262 .020476, .019255, .017477, .016878, .014617, .012489, .011765,
01263 .0099077, .0086446, .0079446, .0078644, .0079763, .008671,
01264 .01001, .0108, .012933, .015349, .016341, .018484, .020254,
01265 .020254, .020478, .019591, .018595, .018385, .019913, .022254,
01266 .024847, .025809, .028053, .029924, .030212, .031367, .03222,
01267 .032739, .032537, .03286, .033344, .033507, .033499, .033339,
01268 .032809, .033041, .031723, .029837, .027511, .026603, .024032,
01269 .021914, .020948, .021701, .023425, .024259, .024987, .023818,
01270 .021768, .019223, .018144, .015282, .012604, .01163, .0097907,
01271 .008336, .0082473, .0079582, .0088077, .009779, .010129, .012145,
01272 .014378, .016761, .01726, .018997, .019998, .019809, .01819,
01273 .016358, .016099, .01617, .017939, .020223, .022521, .02277,
01274 .024279, .025247, .024222, .023989, .023224, .021493, .020362,
01275 .018596, .017309, .015975, .014466, .013171, .011921, .01078,
01276 .0097229, .0087612, .0078729, .0070682, .0063494, .0057156,
01277 .0051459, .0046273, .0041712, .0037686, .0034119, .003095,
01278 .0028126, .0025603, .0023342, .0021314, .0019489, .0017845,
01279 .001636, .0015017, .00138, .0012697, .0011694, .0010782,

```

01280 9.9507e-4, 9.1931e-4, 8.5013e-4, 7.869e-4, 7.2907e-4, 6.7611e-4,  
01281 6.2758e-4, 5.8308e-4, 5.4223e-4, 5.0473e-4, 4.7027e-4, 4.3859e-4,  
01282 4.0946e-4, 3.8265e-4, 3.5798e-4, 3.3526e-4, 3.1436e-4, 2.9511e-4,  
01283 2.7739e-4, 2.6109e-4, 2.4609e-4, 2.3229e-4, 2.1961e-4, 2.0797e-4,  
01284 1.9729e-4, 1.875e-4, 1.7855e-4, 1.7038e-4, 1.6294e-4, 1.5619e-4,  
01285 1.5007e-4, 1.4456e-4, 1.3961e-4, 1.3521e-4, 1.3131e-4, 1.2789e-4,  
01286 1.2494e-4, 1.2242e-4, 1.2032e-4, 1.1863e-4, 1.1733e-4, 1.1641e-4,  
01287 1.1585e-4, 1.1565e-4, 1.158e-4, 1.1629e-4, 1.1712e-4, 1.1827e-4,  
01288 1.1976e-4, 1.2158e-4, 1.2373e-4, 1.262e-4, 1.2901e-4, 1.3214e-4,  
01289 1.3562e-4, 1.3944e-4, 1.4361e-4, 1.4814e-4, 1.5303e-4, 1.5829e-4,  
01290 1.6394e-4, 1.6999e-4, 1.7644e-4, 1.8332e-4, 1.9063e-4, 1.984e-4,  
01291 2.0663e-4, 2.1536e-4, 2.246e-4, 2.3436e-4, 2.4468e-4, 2.5558e-4,  
01292 2.6708e-4, 2.7921e-4, 2.92e-4, 3.0548e-4, 3.1968e-4, 3.3464e-4,  
01293 3.5039e-4, 3.6698e-4, 3.8443e-4, 4.0281e-4, 4.2214e-4, 4.4248e-4,  
01294 4.6389e-4, 4.864e-4, 5.1009e-4, 5.3501e-4, 5.6123e-4, 5.888e-4,  
01295 6.1781e-4, 6.4833e-4, 6.8043e-4, 7.142e-4, 7.4973e-4, 7.8711e-4,  
01296 8.2644e-4, 8.6783e-4, 9.1137e-4, 9.5721e-4, .0010054, .0010562,  
01297 .0011096, .0011659, .0012251, .0012875, .0013532, .0014224,  
01298 .0014953, .001572, .0016529, .0017381, .0018279, .0019226,  
01299 .0020224, .0021277, .0022386, .0023557, .0024792, .0026095,  
01300 .002747, .0028921, .0030453, .0032071, .003378, .0035586,  
01301 .0037494, .003951, .0041642, .0043897, .0046282, .0048805,  
01302 .0051476, .0054304, .00573, .0060473, .0063837, .0067404,  
01303 .0071188, .0075203, .0079466, .0083994, .0088806, .0093922,  
01304 .0099366, .010516, .011134, .011792, .012494, .013244, .014046,  
01305 .014898, .015808, .016781, .017822, .018929, .020108, .02138,  
01306 .022729, .02419, .02576, .027412, .029233, .031198, .033301,  
01307 .035594, .038092, .040767, .04372, .046918, .050246, .053974,  
01308 .058009, .061976, .066586, .071537, .076209, .081856, .087998,  
01309 .093821, .10113, .10913, .11731, .12724, .13821, .15025, .1639,  
01310 .17807, .19472, .21356, .23496, .25758, .28387, .31389, .34104,  
01311 .37469, .40989, .43309, .46845, .5042, .5023, .52981, .55275,  
01312 .51075, .51976, .52457, .44779, .44721, .4503, .4243, .45244,  
01313 .49491, .55399, .39021, .24802, .2501, .2618, .27475, .28879,  
01314 .31317, .33643, .36257, .4018, .43275, .46525, .53333, .56599,  
01315 .60557, .70142, .74194, .77736, .88567, .91182, .93294, .98407,  
01316 .98772, .99176, .9995, 1.2405, 1.3602, 1.338, 1.3255, 1.3267,  
01317 1.3404, 1.3634, 1.3967, 1.4407, 1.4961, 1.5603, 1.6328, 1.7153,  
01318 1.8094, 1.9091, 2.018, 2.1367, 2.264, 2.4035, 2.5562, 2.7179,  
01319 2.9017, 3.1052, 3.3304, 3.5731, 3.8488, 4.1553, 4.4769, 4.7818,  
01320 5.1711, 5.5204, 5.9516, 6.4097, 6.8899, 7.1118, 7.5469, 7.9735,  
01321 7.9511, 8.3014, 8.6418, 8.4757, 8.8256, 9.2294, 9.6923, 10.033,  
01322 10.842, 11.851, 11.78, 8.8435, 9.1381, 9.5956, 10.076, 10.629,  
01323 11.22, 11.883, 12.69, 13.163, 13.974, 14.846, 16.027, 17.053,  
01324 18.148, 19.715, 20.907, 22.163, 23.956, 25.235, 26.566, 27.94,  
01325 29.576, 30.956, 32.432, 35.337, 39.911, 41.128, 42.625, 44.386,  
01326 46.369, 48.619, 51.031, 53.674, 56.825, 59.921, 63.286, 66.929,  
01327 70.859, 75.081, 79.618, 84.513, 89.739, 95.335, 101.35, 107.76,  
01328 114.63, 121.98, 129.87, 138.3, 147.34, 157.04, 167.56, 178.67,  
01329 190.61, 203.43, 217.19, 231.99, 247.88, 264.98, 283.37, 303.17,  
01330 324.49, 347.47, 372.25, 398.98, 427.85, 459.06, 492.8, 529.31,  
01331 568.89, 611.79, 658.35, 708.91, 763.87, 823.65, 888.72, 959.58,  
01332 1036.8, 1121.8, 1213.9, 1314.3, 1423.8, 1543., 1672.8, 1813.4,  
01333 1966.1, 2131.4, 2309.5, 2499.3, 2705., 2925.7, 3161.6, 3411.3,  
01334 3611.5, 3889.2, 4191.1, 4519.3, 4877.9, 5272.9, 5712.9, 6142.7,  
01335 6719.6, 7385., 8145., 8977.7, 9831.9, 10827., 11934., 13063.,  
01336 14434., 15878., 17591., 19435., 21510., 23835., 26835., 29740.,  
01337 32878., 36305., 39830., 43273., 46931., 50499., 49586., 51598.,  
01338 53429., 54619., 55081., 55102., 54485., 53487., 52042., 42689.,  
01339 42607., 44020., 47994., 54169., 53916., 55808., 56642., 46049.,  
01340 44243., 32929., 30658., 21963., 20835., 15962., 13679., 17652.,  
01341 19680., 22388., 25625., 29184., 32520., 35720., 38414., 40523.,  
01342 49228., 48173., 45678., 41768., 37600., 41313., 42654., 44465.,  
01343 55736., 56630., 65409., 63308., 66572., 61845., 60379., 56777.,  
01344 51920., 46601., 41367., 36529., 32219., 28470., 25192., 22362.,  
01345 19907., 1772., 15907., 14273., 12835., 11567., 10445., 9450.2,  
01346 8565.1, 7776., 7070.8, 6439.2, 5872.3, 5362.4, 4903., 4488.3,  
01347 4113.4, 3773.8, 3465.8, 3186.1, 2931.7, 2700.1, 2488.8, 2296.,  
01348 2119.8, 1958.6, 1810.9, 1675.6, 1551.4, 1437.3, 1332.4, 1236.,  
01349 1147.2, 1065.3, 989.86, 920.22, 855.91, 796.48, 741.53, 690.69,  
01350 643.62, 600.02, 559.6, 522.13, 487.35, 455.06, 425.08, 397.21,  
01351 371.3, 347.2, 324.78, 303.9, 284.46, 266.34, 249.45, 233.7,  
01352 219.01, 205.3, 192.5, 180.55, 169.38, 158.95, 149.2, 140.07,  
01353 131.54, 123.56, 116.09, 109.09, 102.54, 96.405, 90.655, 85.266,  
01354 80.213, 75.475, 71.031, 66.861, 62.948, 59.275, 55.827, 52.587,  
01355 49.544, 46.686, 43.998, 41.473, 39.099, 36.867, 34.768, 32.795,  
01356 30.939, 29.192, 27.546, 25.998, 24.539, 23.164, 21.869, 20.65,  
01357 19.501, 18.419, 17.399, 16.438, 15.532, 14.678, 13.874, 13.115,  
01358 12.4, 11.726, 11.088, 10.488, 9.921, 9.3846, 8.8784, 8.3996,  
01359 7.9469, 7.5197, 7.1174, 6.738, 6.379, 6.0409, 5.7213, 5.419,  
01360 5.1327, 4.8611, 4.6046, 4.3617, 4.1316, 3.9138, 3.7077, 3.5125,  
01361 3.3281, 3.1536, 2.9885, 2.8323, 2.6846, 2.5447, 2.4124, 2.2871,  
01362 2.1686, 2.0564, 1.9501, 1.8495, 1.7543, 1.6641, 1.5787, 1.4978,  
01363 1.4212, 1.3486, 1.2799, 1.2147, 1.1529, 1.0943, 1.0388, .98602,  
01364 .93596, .8886, .84352, .80078, .76029, .722, .68585, .65161,  
01365 .61901, .58808, .55854, .53044, .5039, .47853, .45459, .43173,  
01366 .41008, .38965, .37021, .35186, .33444, .31797, .30234, .28758,

```

01367 .2736, .26036, .24764, .2357, .22431, .21342, .20295, .19288,
01368 .18334, .17444, .166, .15815, .15072, .14348, .13674, .13015,
01369 .12399, .11807, .11231, .10689, .10164, .096696, .091955,
01370 .087476, .083183, .079113, .075229, .071536, .068026, .064698,
01371 .06154, .058544, .055699, .052997, .050431, .047993, .045676,
01372 .043475, .041382, .039392, .037501, .035702, .033991, .032364,
01373 .030817, .029345, .027945, .026613, .025345, .024139, .022991,
01374 .021899, .02086, .019871, .018929, .018033, .01718, .016368,
01375 .015595, .014859, .014158, .013491, .012856, .012251, .011675,
01376 .011126, .010604, .010107, .0096331, .009182, .0087523, .0083431,
01377 .0079533, .0075821, .0072284, .0068915, .0065706, .0062649,
01378 .0059737, .0056963, .005432, .0051802, .0049404, .0047118,
01379 .0044941, .0042867, .0040891, .0039009, .0037216, .0035507,
01380 .003388, .0032329, .0030852, .0029445, .0028105, .0026829,
01381 .0025613, .0024455, .0023353, .0022303, .0021304, .0020353,
01382 .0019448, .0018587, .0017767, .0016988, .0016247, .0015543,
01383 .0014874, .0014238, .0013635, .0013062, .0012519, .0012005,
01384 .0011517, .0011057, .0010621, .001021, 9.8233e-4, 9.4589e-4,
01385 9.1167e-4, 8.7961e-4, 8.4964e-4, 8.2173e-4, 7.9582e-4, 7.7189e-4,
01386 7.499e-4, 7.2983e-4, 7.1167e-4, 6.9542e-4, 6.8108e-4, 6.6866e-4,
01387 6.5819e-4, 6.4971e-4, 6.4328e-4, 6.3895e-4, 6.3681e-4, 6.3697e-4,
01388 6.3956e-4, 6.4472e-4, 6.5266e-4, 6.6359e-4, 6.778e-4, 6.9563e-4,
01389 7.1749e-4, 7.4392e-4, 7.7556e-4, 8.1028e-4, 8.4994e-4, 8.8709e-4,
01390 9.3413e-4, 9.6953e-4, .0010202, .0010738, .0010976, .0011507,
01391 .0011686, .0012264, .001291, .0013346, .0014246, .0015293,
01392 .0016359, .0017824, .0019255, .0020854, .002247, .0024148,
01393 .0026199, .0027523, .0029704, .0030702, .0033047, .0035013,
01394 .0037576, .0040275, .0043089, .0046927, .0049307, .0053486,
01395 .0053809, .0056699, .0059325, .0055488, .005634, .0056392,
01396 .004946, .0048855, .0048208, .0044386, .0045498, .0046377,
01397 .0048939, .0052396, .0057324, .0060859, .0066906, .0071148,
01398 .0077224, .0082687, .008769, .0084471, .008572, .0087729,
01399 .008775, .0090742, .0080704, .0080288, .0085747, .0086087,
01400 .0086408, .0088752, .0089381, .0089757, .0093532, .0092824,
01401 .0092566, .0092645, .0092735, .009342, .0095806, .0097991,
01402 .010213, .010611, .011129, .011756, .013237, .01412, .015034,
01403 .015936, .01682, .018597, .019315, .019995, .020658, .021289,
01404 .022363, .022996, .023716, .024512, .025434, .026067, .027118,
01405 .028396, .029865, .031442, .033253, .03525, .037296, .039701,
01406 .042356, .045154, .048059, .051294, .054893, .058636, .061407,
01407 .065172, .068974, .072676, .073379, .076547, .079556, .079134,
01408 .082308, .085739, .090192, .09359, .099599, .10669, .11496,
01409 .1244, .13512, .14752, .14494, .15647, .1668, .17863, .19029,
01410 .20124, .20254, .21179, .21982, .21625, .22364, .23405, .23382,
01411 .2434, .25708, .26406, .27621, .28909, .30395, .31717, .33271,
01412 .3496, .36765, .38774, .40949, .446, .46985, .49846, .5287, .562,
01413 .59841, .64598, .68834, .7327, .78978, .8373, .88708, .94744,
01414 1.0006, 1.0574, 1.1215, 1.1856, 1.2546, 1.3292, 1.4107, 1.4974,
01415 1.5913, 1.6931, 1.8028, 1.9212, 2.0492, 2.1874, 2.3365, 2.4978,
01416 2.6718, 2.8588, 3.062, 3.2818, 3.5188, 3.7752, 4.0527, 4.3542,
01417 4.6782, 5.0312, 5.4123, 5.8246, 6.2639, 6.7435, 7.2636, 7.8064,
01418 8.4091, 9.0696, 9.7677, 10.548, 11.4, 12.309, 13.324, 14.284,
01419 15.445, 16.687, 18.019, 19.403, 20.847, 22.366, 23.925, 25.537,
01420 27.213, 28.069, 29.864, 31.829, 33.988, 35.856, 38.829, 42.321,
01421 46.319, 50.606, 55.126, 59.126, 64.162, 68.708, 74.615, 81.176,
01422 87.739, 95.494, 103.83, 113.38, 123.99, 135.8, 148.7, 162.58,
01423 176.32, 192.6, 211.47, 232.7, 252.64, 277.41, 305.38, 333.44,
01424 366.42, 402.66, 442.14, 484.53, 526.42, 568.15, 558.78, 582.6,
01425 600.98, 613.94, 619.44, 618.24, 609.84, 595.96, 484.86, 475.59,
01426 478.49, 501.56, 552.19, 628.44, 630.39, 658.92, 671.96, 562.7,
01427 545.88, 423.43, 400.14, 306.59, 294.13, 246.8, 226.51, 278.21,
01428 314.39, 347.22, 389.13, 433.16, 477.48, 521.67, 560.54, 683.6,
01429 696.37, 695.91, 683.1, 658.24, 634.89, 698.85, 742.87, 796.66,
01430 954.49, 1009.5, 1150.5, 1179.1, 1267.9, 1272.4, 1312.7, 1330.4,
01431 1331.6, 1315.8, 1308.3, 1293.3, 1274.6, 1249.5, 1213.2, 1172.1,
01432 1124.4, 930.33, 893.36, 871.27, 883.54, 940.76, 1036., 1025.6,
01433 1053.1, 914.51, 894.15, 865.03, 670.63, 508.41, 475.15, 370.85,
01434 361.06, 319.38, 312.75, 331.87, 367.13, 415., 467.94, 525.49,
01435 578.41, 624.66, 794.82, 796.97, 780.29, 736.49, 670.18, 603.75,
01436 659.67, 679.8, 857.12, 884.05, 900.65, 1046.1, 1141.9, 1083.,
01437 1089.2, 1e3, 947.08, 872.31, 787.91, 704.75, 624.93, 553.68,
01438 489.91, 434.21, 385.64, 343.3, 306.42, 274.18, 245.94, 221.11,
01439 199.23, 179.88, 162.73, 147.48, 133.88, 121.73, 110.86, 101.1,
01440 92.323, 84.417, 77.281, 70.831, 64.991, 59.694, 54.884, 50.509,
01441 46.526, 42.893, 39.58, 36.549, 33.776, 31.236, 28.907, 26.77,
01442 24.805, 23., 21.339, 19.81, 18.404, 17.105, 15.909, 14.801,
01443 13.778, 12.83, 11.954, 11.142, 10.389, 9.691, 9.0434, 8.4423,
01444 7.8842, 7.3657, 6.8838, 6.4357, 6.0189, 5.6308, 5.2696, 4.9332,
01445 4.6198, 4.3277, 4.0553, 3.8012, 3.5639, 3.3424, 3.1355, 2.9422,
01446 2.7614, 2.5924, 2.4343, 2.2864, 2.148, 2.0184, 1.8971, 1.7835,
01447 1.677, 1.5773, 1.4838, 1.3961, 1.3139, 1.2369, 1.1645, 1.0966,
01448 1.0329, .97309, .91686, .86406, .81439, .76767, .72381, .68252,
01449 .64359, .60695, .57247, .54008, .50957, .48092, .45401, .42862,
01450 .40465, .38202, .36072, .34052, .3216, .30386, .28711, .27135,
01451 .25651, .24252, .2293, .21689, .20517, .19416, .18381, .17396,
01452 .16469
01453 };

```



```
01454
01455 static double co2230[2001] = { 2.743e-5, 2.8815e-5, 3.027e-5, 3.1798e-5,
01456 3.3405e-5, 3.5094e-5, 3.6869e-5, 3.8734e-5, 4.0694e-5, 4.2754e-5,
01457 4.492e-5, 4.7196e-5, 4.9588e-5, 5.2103e-5, 5.4747e-5, 5.7525e-5,
01458 6.0446e-5, 6.3516e-5, 6.6744e-5, 7.0137e-5, 7.3704e-5, 7.7455e-5,
01459 8.1397e-5, 8.5543e-5, 8.9901e-5, 9.4484e-5, 9.9302e-5, 1.0437e-4,
01460 1.097e-4, 1.153e-4, 1.2119e-4, 1.2738e-4, 1.3389e-4, 1.4074e-4,
01461 1.4795e-4, 1.5552e-4, 1.6349e-4, 1.7187e-4, 1.8068e-4, 1.8995e-4,
01462 1.997e-4, 2.0996e-4, 2.2075e-4, 2.321e-4, 2.4403e-4, 2.5659e-4,
01463 2.698e-4, 2.837e-4, 2.9832e-4, 3.137e-4, 3.2988e-4, 3.4691e-4,
01464 3.6483e-4, 3.8368e-4, 4.0351e-4, 4.2439e-4, 4.4635e-4, 4.6947e-4,
01465 4.9379e-4, 5.1939e-4, 5.4633e-4, 5.7468e-4, 6.0452e-4, 6.3593e-4,
01466 6.69e-4, 7.038e-4, 7.4043e-4, 7.79e-4, 8.1959e-4, 8.6233e-4,
01467 9.0732e-4, 9.5469e-4, .0010046, .0010571, .0011124, .0011706,
01468 .0012319, .0012964, .0013644, .001436, .0015114, .0015908,
01469 .0016745, .0017625, .0018553, .0019531, .002056, .0021645,
01470 .0022788, .0023992, .002526, .0026596, .0028004, .0029488,
01471 .0031052, .0032699, .0034436, .0036265, .0038194, .0040227,
01472 .0042369, .0044628, .0047008, .0049518, .0052164, .0054953,
01473 .0057894, .0060995, .0064265, .0067713, .007135, .0075184,
01474 .0079228, .0083494, .0087993, .0092738, .0097745, .010303,
01475 .01086, .011448, .012068, .012722, .013413, .014142, .014911,
01476 .015723, .01658, .017484, .018439, .019447, .020511, .021635,
01477 .022821, .024074, .025397, .026794, .02827, .029829, .031475,
01478 .033215, .035052, .036994, .039045, .041213, .043504, .045926,
01479 .048485, .05119, .05405, .057074, .060271, .063651, .067225,
01480 .071006, .075004, .079233, .083708, .088441, .093449, .098749,
01481 .10436, .11029, .11657, .12322, .13026, .13772, .14561, .15397,
01482 .16282, .1722, .18214, .19266, .20381, .21563, .22816, .24143,
01483 .2555, .27043, .28625, .30303, .32082, .3397, .35972, .38097,
01484 .40352, .42746, .45286, .47983, .50847, .53888, .57119, .6055,
01485 .64196, .6807, .72187, .76564, .81217, .86165, .91427, .97025,
01486 1.0298, 1.0932, 1.1606, 1.2324, 1.3088, 1.3902, 1.477, 1.5693,
01487 1.6678, 1.7727, 1.8845, 2.0038, 2.131, 2.2666, 2.4114, 2.5659,
01488 2.7309, 2.907, 3.0951, 3.2961, 3.5109, 3.7405, 3.986, 4.2485,
01489 4.5293, 4.8299, 5.1516, 5.4961, 5.8651, 6.2605, 6.6842, 7.1385,
01490 7.6256, 8.1481, 8.7089, 9.3109, 9.9573, 10.652, 11.398, 12.2,
01491 13.063, 13.992, 14.99, 16.064, 17.222, 18.469, 19.813, 21.263,
01492 22.828, 24.516, 26.34, 28.31, 30.437, 32.738, 35.226, 37.914,
01493 40.824, 43.974, 47.377, 51.061, 55.011, 59.299, 63.961, 69.013,
01494 74.492, 80.444, 86.919, 93.836, 101.23, 109.25, 117.98, 127.47,
01495 137.81, 149.07, 161.35, 174.75, 189.42, 205.49, 223.02, 242.26,
01496 263.45, 286.75, 311.94, 340.01, 370.86, 404.92, 440.44, 480.27,
01497 525.17, 574.71, 626.22, 686.8, 754.38, 827.07, 913.38, 1011.7,
01498 1121.5, 1161.6, 1289.5, 1432.2, 1595.4, 1777., 1983.3, 2216.1,
01499 2485.7, 2788.3, 3101.5, 3481., 3902.1, 4257.1, 4740., 5272.8,
01500 5457.9, 5946.2, 6505.3, 6668.4, 7302.4, 8061.6, 9015.8, 9908.3,
01501 11613., 13956., 3249.6, 3243., 2901.5, 2841.3, 2729.6, 2558.2,
01502 1797.8, 1583.2, 1386., 1233.5, 787.74, 701.46, 761.66, 767.21,
01503 722.83, 1180.6, 1332.1, 1461.6, 2032.9, 2166., 2255.9, 2294.7,
01504 2587.2, 2396.5, 2122.4, 12553., 10784., 9832.5, 8827.3, 8029.1,
01505 7377.9, 7347.1, 6783.8, 6239.1, 5721.1, 5503., 4975.1, 4477.8,
01506 4021.3, 3676.8, 3275.3, 2914.9, 2597.4, 2328.2, 2075.4, 1857.6,
01507 1663.6, 1493.3, 1343.8, 1213.3, 1095.6, 1066.5, 958.91, 865.15,
01508 783.31, 714.35, 650.77, 593.98, 546.2, 499.9, 457.87, 421.75,
01509 387.61, 355.25, 326.62, 299.7, 275.21, 253.17, 232.83, 214.31,
01510 197.5, 182.08, 167.98, 155.12, 143.32, 132.5, 122.58, 113.48,
01511 105.11, 97.415, 90.182, 83.463, 77.281, 71.587, 66.341, 61.493,
01512 57.014, 53.062, 49.21, 45.663, 42.38, 39.348, 36.547, 33.967,
01513 31.573, 29.357, 27.314, 25.415, 23.658, 22.03, 20.524, 19.125,
01514 17.829, 16.627, 15.511, 14.476, 13.514, 12.618, 11.786, 11.013,
01515 10.294, 9.6246, 9.0018, 8.4218, 7.8816, 7.3783, 6.9092, 6.4719,
01516 6.0641, 5.6838, 5.3289, 4.998, 4.6893, 4.4014, 4.1325, 3.8813,
01517 3.6469, 3.4283, 3.2241, 3.035, 2.8576, 2.6922, 2.5348, 2.3896,
01518 2.2535, 2.1258, 2.0059, 1.8929, 1.7862, 1.6854, 1.5898, 1.4992,
01519 1.4017, 1.3218, 1.2479, 1.1809, 1.1215, 1.0693, 1.0116, .96016,
01520 .9105, .84859, .80105, .74381, .69982, .65127, .60899, .57843,
01521 .54592, .51792, .49336, .47155, .45201, .43426, .41807, .40303,
01522 .38876, .3863, .37098, .35492, .33801, .32032, .30341, .29874,
01523 .29193, .28689, .29584, .29155, .29826, .29195, .29287, .2904,
01524 .28199, .27709, .27162, .26622, .26133, .25676, .25235, .23137,
01525 .22365, .21519, .20597, .19636, .18699, .16485, .16262, .16643,
01526 .17542, .18198, .18631, .16759, .16338, .13505, .1267, .10053,
01527 .092554, .074093, .062159, .055523, .054849, .05401, .05528,
01528 .058982, .07952, .08647, .093244, .099285, .10393, .10661,
01529 .12072, .11417, .10396, .093265, .089137, .088909, .10902,
01530 .11277, .13625, .13565, .14907, .14167, .1428, .13744, .12768,
01531 .11382, .10244, .091686, .08109, .071739, .063616, .056579,
01532 .050504, .045251, .040689, .036715, .033237, .030181, .027488,
01533 .025107, .022998, .021125, .01946, .017979, .016661, .015489,
01534 .014448, .013526, .012712, .011998, .011375, .010839, .010384,
01535 .010007, .0097053, .0094783, .0093257, .0092489, .0092504,
01536 .0093346, .0095077, .0097676, .01012, .01058, .011157, .011844,
01537 .012672, .013665, .014766, .015999, .017509, .018972, .020444,
01538 .022311, .023742, .0249, .025599, .026981, .026462, .025143,
01539 .025066, .022814, .020458, .020026, .019142, .020189, .022371,
01540 .024163, .023728, .02199, .019506, .018591, .015576, .012784,
```

```
01541 .011744, .0094777, .0079148, .0070652, .006986, .0071758,
01542 .008086, .0098025, .01087, .013609, .016764, .018137, .021061,
01543 .023498, .023576, .023965, .022828, .021519, .021283, .023364,
01544 .026457, .029782, .030856, .033486, .035515, .035543, .036558,
01545 .037198, .037472, .037045, .037284, .03777, .038085, .038366,
01546 .038526, .038282, .038915, .037697, .035667, .032941, .031959,
01547 .028692, .025918, .024596, .025592, .027873, .028935, .02984,
01548 .028148, .025305, .021912, .020454, .016732, .013357, .01205,
01549 .009731, .0079881, .0077704, .0074387, .0083895, .0096776,
01550 .010326, .01293, .015955, .019247, .020145, .02267, .024231,
01551 .024184, .022131, .019784, .01955, .01971, .022119, .025116,
01552 .027978, .028107, .029808, .030701, .029164, .028551, .027286,
01553 .024946, .023259, .020982, .019221, .017471, .015643, .014074,
01554 .01261, .011301, .010116, .0090582, .0081036, .0072542, .0065034,
01555 .0058436, .0052571, .0047321, .0042697, .0038607, .0034977,
01556 .0031747, .0028864, .0026284, .002397, .002189, .0020017,
01557 .0018326, .0016798, .0015414, .0014159, .0013019, .0011983,
01558 .0011039, .0010177, 9.391e-4, 8.6717e-4, 8.0131e-4, 7.4093e-4,
01559 6.8553e-4, 6.3464e-4, 5.8787e-4, 5.4487e-4, 5.0533e-4, 4.69e-4,
01560 4.3556e-4, 4.0474e-4, 3.7629e-4, 3.5e-4, 3.2569e-4, 3.032e-4,
01561 2.8239e-4, 2.6314e-4, 2.4535e-4, 2.2891e-4, 2.1374e-4, 1.9975e-4,
01562 1.8685e-4, 1.7498e-4, 1.6406e-4, 1.5401e-4, 1.4479e-4, 1.3633e-4,
01563 1.2858e-4, 1.2148e-4, 1.1499e-4, 1.0907e-4, 1.0369e-4, 9.8791e-5,
01564 9.4359e-5, 9.0359e-5, 8.6766e-5, 8.3555e-5, 8.0703e-5, 7.8192e-5,
01565 7.6003e-5, 7.4119e-5, 7.2528e-5, 7.1216e-5, 7.0171e-5, 6.9385e-5,
01566 6.8848e-5, 6.8554e-5, 6.8496e-5, 6.8669e-5, 6.9069e-5, 6.9694e-5,
01567 7.054e-5, 7.1608e-5, 7.2896e-5, 7.4406e-5, 7.6139e-5, 7.8097e-5,
01568 8.0283e-5, 8.2702e-5, 8.5357e-5, 8.8255e-5, 9.1402e-5, 9.4806e-5,
01569 9.8473e-5, 1.0241e-4, 1.0664e-4, 1.1115e-4, 1.1598e-4, 1.2112e-4,
01570 1.2659e-4, 1.3241e-4, 1.3859e-4, 1.4515e-4, 1.521e-4, 1.5947e-4,
01571 1.6728e-4, 1.7555e-4, 1.8429e-4, 1.9355e-4, 2.0334e-4, 2.1369e-4,
01572 2.2463e-4, 2.3619e-4, 2.4841e-4, 2.6132e-4, 2.7497e-4, 2.8938e-4,
01573 3.0462e-4, 3.2071e-4, 3.3771e-4, 3.5567e-4, 3.7465e-4, 3.947e-4,
01574 4.1588e-4, 4.3828e-4, 4.6194e-4, 4.8695e-4, 5.1338e-4, 5.4133e-4,
01575 5.7087e-4, 6.0211e-4, 6.3515e-4, 6.701e-4, 7.0706e-4, 7.4617e-4,
01576 7.8756e-4, 8.3136e-4, 8.7772e-4, 9.2681e-4, 9.788e-4, .0010339,
01577 .0010922, .001154, .0012195, .0012889, .0013626, .0014407,
01578 .0015235, .0016114, .0017048, .0018038, .001909, .0020207,
01579 .0021395, .0022657, .0023998, .0025426, .0026944, .002856,
01580 .0030281, .0032114, .0034068, .003615, .0038371, .004074,
01581 .004327, .0045971, .0048857, .0051942, .0055239, .0058766,
01582 .0062538, .0066573, .0070891, .007551, .0080455, .0085747,
01583 .0091412, .0097481, .010397, .011092, .011837, .012638, .013495,
01584 .014415, .01541, .016475, .017621, .018857, .020175, .02162,
01585 .023185, .024876, .02672, .028732, .030916, .033319, .035939,
01586 .038736, .041847, .04524, .048715, .052678, .056977, .061203,
01587 .066184, .07164, .076952, .083477, .090674, .098049, .10697,
01588 .1169, .1277, .14011, .15323, .1684, .18601, .20626, .22831,
01589 .25417, .28407, .31405, .34957, .38823, .41923, .46026, .50409,
01590 .51227, .54805, .57976, .53818, .55056, .557, .46741, .46403,
01591 .4636, .42265, .45166, .49852, .56663, .34306, .17779, .17697,
01592 .18346, .19129, .20014, .21778, .23604, .25649, .28676, .31238,
01593 .33856, .39998, .4288, .46568, .56654, .60786, .64473, .76466,
01594 .7897, .80778, .86443, .85736, .84798, .84157, 1.1385, 1.2446,
01595 1.1923, 1.1552, 1.1338, 1.1266, 1.1292, 1.1431, 1.1683, 1.2059,
01596 1.2521, 1.3069, 1.3712, 1.4471, 1.5275, 1.6165, 1.7145, 1.8189,
01597 1.9359, 2.065, 2.2007, 2.3591, 2.5362, 2.7346, 2.9515, 3.2021,
01598 3.4851, 3.7935, 4.0694, 4.4463, 4.807, 5.2443, 5.7178, 6.2231,
01599 6.4796, 6.9461, 7.4099, 7.3652, 7.7182, 8.048, 7.7373, 8.0363,
01600 8.3855, 8.8044, 9.0257, 9.8574, 10.948, 10.563, 6.8979, 7.0744,
01601 7.4121, 7.7663, 8.1768, 8.6243, 9.1437, 9.7847, 10.182, 10.849,
01602 11.572, 12.602, 13.482, 14.431, 15.907, 16.983, 18.11, 19.884,
01603 21.02, 22.18, 23.355, 24.848, 25.954, 27.13, 30.186, 34.893,
01604 35.682, 36.755, 38.111, 39.703, 41.58, 43.606, 45.868, 48.573,
01605 51.298, 54.291, 57.559, 61.116, 64.964, 69.124, 73.628, 78.471,
01606 83.683, 89.307, 95.341, 101.84, 108.83, 116.36, 124.46, 133.18,
01607 142.57, 152.79, 163.69, 175.43, 188.11, 201.79, 216.55, 232.51,
01608 249.74, 268.38, 288.54, 310.35, 333.97, 359.55, 387.26, 417.3,
01609 449.88, 485.2, 523.54, 565.14, 610.28, 659.31, 712.56, 770.43,
01610 833.36, 901.82, 976.36, 1057.6, 1146.8, 1243.8, 1350, 1466.3,
01611 1593.6, 1732.7, 1884.1, 2049.1, 2228.2, 2421.9, 2629.4, 2853.7,
01612 3094.4, 3351.1, 3622.3, 3829.8, 4123.1, 4438.3, 4777.2, 5144.1,
01613 5545.4, 5990.5, 6404.5, 6996.8, 7687.6, 8482.9, 9349.4, 10203,
01614 11223, 12358, 13493, 14916, 16416, 18236, 20222, 22501,
01615 25102, 28358, 31707, 35404, 39538, 43911, 48391, 53193,
01616 58028, 58082, 61276, 64193, 66294, 67480, 67921, 67423,
01617 66254, 64341, 51737, 51420, 53072, 58145, 66195, 65358,
01618 67377, 67869, 53509, 50553, 35737, 32425, 21704, 19974,
01619 14457, 12142, 16798, 19489, 23049, 27270, 31910, 36457,
01620 40877, 44748, 47876, 59793, 58626, 55454, 50337, 44893,
01621 50228, 52216, 54747, 69541, 70455, 81014, 77694, 80533,
01622 73953, 70927, 65539, 59002, 52281, 45953, 40292, 35360,
01623 31124, 27478, 24346, 21647, 19308, 17271, 15491, 13927,
01624 12550, 11331, 10250, 9288, 8431.4, 7664.9, 6978.3, 6361.8,
01625 5807.4, 5307.7, 4856.8, 4449, 4079.8, 3744.9, 3440.8, 3164.2,
01626 2912.3, 2682.7, 2473, 2281.4, 2106, 1945.3, 1797.9, 1662.5,
01627 1538.1, 1423.6, 1318.1, 1221, 1131.5, 1049, 972.99, 902.87,
```

01628 838.01, 777.95, 722.2, 670.44, 622.35, 577.68, 536.21, 497.76,  
01629 462.12, 429.13, 398.61, 370.39, 344.29, 320.16, 297.85, 277.2,  
01630 258.08, 240.38, 223.97, 208.77, 194.66, 181.58, 169.43, 158.15,  
01631 147.67, 137.92, 128.86, 120.44, 112.6, 105.3, 98.499, 92.166,  
01632 86.264, 80.763, 75.632, 70.846, 66.381, 62.213, 58.321, 54.685,  
01633 51.288, 48.114, 45.145, 42.368, 39.772, 37.341, 35.065, 32.937,  
01634 30.943, 29.077, 27.33, 25.693, 24.158, 22.717, 21.367, 20.099,  
01635 18.909, 17.792, 16.744, 15.761, 14.838, 13.971, 13.157, 12.393,  
01636 11.676, 11.003, 10.369, 9.775, 9.2165, 8.6902, 8.1963, 7.7314,  
01637 7.2923, 6.8794, 6.4898, 6.122, 5.7764, 5.4525, 5.1484, 4.8611,  
01638 4.5918, 4.3379, 4.0982, 3.8716, 3.6567, 3.4545, 3.2634, 3.0828,  
01639 2.9122, 2.7512, 2.5993, 2.4561, 2.3211, 2.1938, 2.0737, 1.9603,  
01640 1.8534, 1.7525, 1.6572, 1.5673, 1.4824, 1.4022, 1.3265, 1.2551,  
01641 1.1876, 1.1239, 1.0637, 1.0069, .9532, .90248, .85454, .80921,  
01642 .76631, .72569, .6872, .65072, .61635, .5836, .55261, .52336,  
01643 .49581, .46998, .44559, .42236, .40036, .37929, .35924, .34043,  
01644 .32238, .30547, .28931, .27405, .25975, .24616, .23341, .22133,  
01645 .20997, .19924, .18917, .17967, .17075, .16211, .15411, .14646,  
01646 .13912, .13201, .12509, .11857, .11261, .10698, .10186, .097039,  
01647 .092236, .087844, .083443, .07938, .075452, .071564, .067931,  
01648 .064389, .061078, .057901, .054921, .052061, .049364, .046789,  
01649 .04435, .042044, .039866, .037808, .035863, .034023, .032282,  
01650 .030634, .029073, .027595, .026194, .024866, .023608, .022415,  
01651 .021283, .02021, .019193, .018228, .017312, .016443, .015619,  
01652 .014837, .014094, .01339, .012721, .012086, .011483, .010911,  
01653 .010368, .009852, .0093623, .0088972, .0084556, .0080362,  
01654 .0076379, .0072596, .0069003, .006559, .0062349, .0059269,  
01655 .0056344, .0053565, .0050925, .0048417, .0046034, .004377,  
01656 .0041618, .0039575, .0037633, .0035788, .0034034, .0032368,  
01657 .0030785, .002928, .0027851, .0026492, .0025201, .0023975,  
01658 .0022809, .0021701, .0020649, .0019649, .0018699, .0017796,  
01659 .0016938, .0016122, .0015348, .0014612, .0013913, .001325,  
01660 .0012619, .0012021, .0011452, .0010913, .0010401, 9.9149e-4,  
01661 9.454e-4, 9.0169e-4, 8.6024e-4, 8.2097e-4, 7.8377e-4, 7.4854e-4,  
01662 7.1522e-4, 6.8371e-4, 6.5393e-4, 6.2582e-4, 5.9932e-4, 5.7435e-4,  
01663 5.5087e-4, 5.2882e-4, 5.0814e-4, 4.8881e-4, 4.7076e-4, 4.5398e-4,  
01664 4.3843e-4, 4.2407e-4, 4.109e-4, 3.9888e-4, 3.88e-4, 3.7826e-4,  
01665 3.6963e-4, 3.6213e-4, 3.5575e-4, 3.505e-4, 3.464e-4, 3.4346e-4,  
01666 3.4173e-4, 3.4125e-4, 3.4206e-4, 3.4424e-4, 3.4787e-4, 3.5303e-4,  
01667 3.5986e-4, 3.6847e-4, 3.7903e-4, 3.9174e-4, 4.0681e-4, 4.2455e-4,  
01668 4.4527e-4, 4.6942e-4, 4.9637e-4, 5.2698e-4, 5.5808e-4, 5.9514e-4,  
01669 6.2757e-4, 6.689e-4, 7.1298e-4, 7.3955e-4, 7.8403e-4, 8.0449e-4,  
01670 8.5131e-4, 9.0256e-4, 9.3692e-4, .0010051, .0010846, .0011678,  
01671 .001282, .0014016, .0015355, .0016764, .0018272, .0020055,  
01672 .0021455, .0023421, .0024615, .0026786, .0028787, .0031259,  
01673 .0034046, .0036985, .0040917, .0043902, .0048349, .0049531,  
01674 .0052989, .0056148, .0052452, .0053357, .005333, .0045069,  
01675 .0043851, .004253, .003738, .0038084, .0039013, .0041505,  
01676 .0045372, .0050569, .0054507, .0061267, .0066122, .0072449,  
01677 .0078012, .0082651, .0076538, .0076573, .0076806, .0075227,  
01678 .0076269, .0063758, .006254, .0067749, .0067909, .0068231,  
01679 .0072143, .0072762, .0072954, .007679, .0075107, .0073658,  
01680 .0072441, .0071074, .0070378, .007176, .0072472, .0075844,  
01681 .0079291, .008412, .0090165, .010688, .011535, .012375, .013166,  
01682 .013895, .015567, .016011, .016392, .016737, .017043, .017731,  
01683 .018031, .018419, .018877, .019474, .019868, .020604, .021538,  
01684 .022653, .023869, .025288, .026879, .028547, .030524, .03274,  
01685 .035132, .03769, .040567, .043793, .047188, .049962, .053542,  
01686 .057205, .060776, .061489, .064419, .067124, .065945, .068487,  
01687 .071209, .074783, .077039, .082444, .08902, .09692, .10617,  
01688 .11687, .12952, .12362, .13498, .14412, .15492, .16519, .1744,  
01689 .17096, .17714, .18208, .17363, .17813, .18564, .18295, .19045,  
01690 .20252, .20815, .21844, .22929, .24229, .25321, .26588, .2797,  
01691 .29465, .31136, .32961, .36529, .38486, .41027, .43694, .4667,  
01692 .49943, .54542, .58348, .62303, .67633, .71755, .76054, .81371,  
01693 .85934, .90841, .96438, 1.0207, 1.0821, 1.1491, 1.2226, 1.3018,  
01694 1.388, 1.4818, 1.5835, 1.6939, 1.8137, 1.9435, 2.0843, 2.237,  
01695 2.4026, 2.5818, 2.7767, 2.9885, 3.2182, 3.4679, 3.7391, 4.0349,  
01696 4.3554, 4.7053, 5.0849, 5.4986, 5.9436, 6.4294, 6.9598, 7.5203,  
01697 8.143, 8.8253, 9.5568, 10.371, 11.267, 12.233, 13.31, 14.357,  
01698 15.598, 16.93, 18.358, 19.849, 21.408, 23.04, 24.706, 26.409,  
01699 28.153, 28.795, 30.549, 32.43, 34.49, 36.027, 38.955, 42.465,  
01700 46.565, 50.875, 55.378, 59.002, 63.882, 67.949, 73.693, 80.095,  
01701 86.403, 94.264, 102.65, 112.37, 123.3, 135.54, 149.14, 163.83,  
01702 179.17, 196.89, 217.91, 240.94, 264.13, 292.39, 324.83, 358.21,  
01703 397.16, 440.5, 488.6, 541.04, 595.3, 650.43, 652.03, 688.74,  
01704 719.47, 743.54, 757.68, 762.35, 756.43, 741.42, 595.43, 580.97,  
01705 580.83, 605.68, 667.88, 764.49, 759.93, 789.12, 798.17, 645.66,  
01706 615.65, 455.05, 421.09, 306.45, 289.14, 235.7, 215.52, 274.57,  
01707 316.53, 357.73, 409.89, 465.06, 521.84, 579.02, 630.64, 794.46,  
01708 813., 813.56, 796.25, 761.57, 727.97, 812.14, 866.75, 932.5,  
01709 1132.8, 1194.8, 1362.2, 1387.2, 1482.3, 1479.7, 1517.9, 1533.1,  
01710 1534.2, 1523.3, 1522.5, 1515.5, 1505.2, 1486.5, 1454., 1412.,  
01711 1358.8, 1107.8, 1060.9, 1033.5, 1048.2, 1122.4, 1248.9, 1227.1,  
01712 1255.4, 1058.9, 1020.7, 970.59, 715.24, 512.56, 468.47, 349.3,  
01713 338.26, 299.22, 301.26, 332.38, 382.08, 445.49, 515.87, 590.85,  
01714 662.3, 726.05, 955.59, 964.11, 945.17, 891.48, 807.11, 720.9,

```

01715      803.36, 834.46, 1073.9, 1107.1, 1123.6, 1296., 1393.7, 1303.1,
01716      1284.3, 1161.8, 1078.8, 976.13, 868.72, 767.4, 674.72, 593.73,
01717      523.12, 462.24, 409.75, 364.34, 325., 290.73, 260.76, 234.46,
01718      211.28, 190.78, 172.61, 156.44, 142.01, 129.12, 117.57, 107.2,
01719      97.877, 89.47, 81.882, 75.021, 68.807, 63.171, 58.052, 53.396,
01720      49.155, 45.288, 41.759, 38.531, 35.576, 32.868, 30.384, 28.102,
01721      26.003, 24.071, 22.293, 20.655, 19.147, 17.756, 16.476, 15.292,
01722      14.198, 13.183, 12.241, 11.367, 10.554, 9.7989, 9.0978, 8.4475,
01723      7.845, 7.2868, 6.7704, 6.2927, 5.8508, 5.4421, 5.064, 4.714,
01724      4.3902, 4.0902, 3.8121, 3.5543, 3.315, 3.093, 2.8869, 2.6953,
01725      2.5172, 2.3517, 2.1977, 2.0544, 1.9211, 1.7969, 1.6812, 1.5735,
01726      1.4731, 1.3794, 1.2921, 1.2107, 1.1346, 1.0637, .99744, .93554,
01727      .87771, .82368, .77313, .72587, .6816, .64014, .60134, .565,
01728      .53086, .49883, .46881, .44074, .4144, .38979, .36679, .34513,
01729      .32474, .30552, .28751, .27045, .25458, .23976, .22584, .21278,
01730      .20051, .18899, .17815, .16801, .15846, .14954, .14117, .13328,
01731      .12584
01732  };
01733
01734  /* Get CO2 continuum absorption... */
01735  double xw = nu / 2 + 1;
01736  if (xw >= 1 && xw < 2001) {
01737      int iw = (int) xw;
01738      double dw = xw - iw;
01739      double ew = 1 - dw;
01740      double cw296 = ew * co2296[iw - 1] + dw * co2296[iw];
01741      double cw260 = ew * co2260[iw - 1] + dw * co2260[iw];
01742      double cw230 = ew * co2230[iw - 1] + dw * co2230[iw];
01743      double dt230 = t - 230;
01744      double dt260 = t - 260;
01745      double dt296 = t - 296;
01746      double ctw = dt260 * 5.050505e-4 * dt296 * cw230 - dt230 * 9.259259e-4
01747          * dt296 * cw260 + dt230 * 4.208754e-4 * dt260 * cw296;
01748      return u / NA / 1000 * p / P0 * ctw;
01749  } else
01750      return 0;
01751 }

```

### 5.17.2.7 ctmh2o() double ctmh2o (

```

double nu,
double p,
double t,
double q,
double u )

```

Compute water vapor continuum (optical depth).

Definition at line 1755 of file [jurassic.c](#).

```

01760      {
01761
01762      static double h2o296[2001] = { .17, .1695, .172, .168, .1687, .1624, .1606,
01763      .1508, .1447, .1344, .1214, .1133, .1009, .09217, .08297, .06989,
01764      .06513, .05469, .05056, .04417, .03779, .03484, .02994, .0272,
01765      .02325, .02063, .01818, .01592, .01405, .01251, .0108, .009647,
01766      .008424, .007519, .006555, .00588, .005136, .004511, .003989,
01767      .003509, .003114, .00274, .002446, .002144, .001895, .001676,
01768      .001486, .001312, .001164, .001031, 9.129e-4, 8.106e-4, 7.213e-4,
01769      6.4e-4, 5.687e-4, 5.063e-4, 4.511e-4, 4.029e-4, 3.596e-4,
01770      3.22e-4, 2.889e-4, 2.597e-4, 2.337e-4, 2.108e-4, 1.907e-4,
01771      1.728e-4, 1.57e-4, 1.43e-4, 1.305e-4, 1.195e-4, 1.097e-4,
01772      1.009e-4, 9.307e-5, 8.604e-5, 7.971e-5, 7.407e-5, 6.896e-5,
01773      6.433e-5, 6.013e-5, 5.631e-5, 5.283e-5, 4.963e-5, 4.669e-5,
01774      4.398e-5, 4.148e-5, 3.917e-5, 3.702e-5, 3.502e-5, 3.316e-5,
01775      3.142e-5, 2.978e-5, 2.825e-5, 2.681e-5, 2.546e-5, 2.419e-5,
01776      2.299e-5, 2.186e-5, 2.079e-5, 1.979e-5, 1.884e-5, 1.795e-5,
01777      1.711e-5, 1.633e-5, 1.559e-5, 1.49e-5, 1.426e-5, 1.367e-5,
01778      1.312e-5, 1.263e-5, 1.218e-5, 1.178e-5, 1.143e-5, 1.112e-5,
01779      1.088e-5, 1.07e-5, 1.057e-5, 1.05e-5, 1.051e-5, 1.059e-5,
01780      1.076e-5, 1.1e-5, 1.133e-5, 1.18e-5, 1.237e-5, 1.308e-5,
01781      1.393e-5, 1.483e-5, 1.614e-5, 1.758e-5, 1.93e-5, 2.123e-5,
01782      2.346e-5, 2.647e-5, 2.93e-5, 3.279e-5, 3.745e-5, 4.152e-5,
01783      4.813e-5, 5.477e-5, 6.203e-5, 7.331e-5, 8.056e-5, 9.882e-5,
01784      1.05e-4, 1.21e-4, 1.341e-4, 1.572e-4, 1.698e-4, 1.968e-4,
01785      2.175e-4, 2.431e-4, 2.735e-4, 2.867e-4, 3.19e-4, 3.371e-4,
01786      3.554e-4, 3.726e-4, 3.837e-4, 3.878e-4, 3.864e-4, 3.858e-4,
01787      3.841e-4, 3.852e-4, 3.815e-4, 3.762e-4, 3.618e-4, 3.579e-4,
01788      3.45e-4, 3.202e-4, 3.018e-4, 2.785e-4, 2.602e-4, 2.416e-4,

```

01789 2.097e-4, 1.939e-4, 1.689e-4, 1.498e-4, 1.308e-4, 1.17e-4,  
01790 1.011e-4, 9.237e-5, 7.909e-5, 7.006e-5, 6.112e-5, 5.401e-5,  
01791 4.914e-5, 4.266e-5, 3.963e-5, 3.316e-5, 3.037e-5, 2.598e-5,  
01792 2.294e-5, 2.066e-5, 1.813e-5, 1.583e-5, 1.423e-5, 1.247e-5,  
01793 1.116e-5, 9.76e-6, 8.596e-6, 7.72e-6, 6.825e-6, 6.108e-6,  
01794 5.366e-6, 4.733e-6, 4.229e-6, 3.731e-6, 3.346e-6, 2.972e-6,  
01795 2.628e-6, 2.356e-6, 2.102e-6, 1.878e-6, 1.678e-6, 1.507e-6,  
01796 1.348e-6, 1.21e-6, 1.089e-6, 9.806e-7, 8.857e-7, 8.004e-7,  
01797 7.261e-7, 6.599e-7, 6.005e-7, 5.479e-7, 5.011e-7, 4.595e-7,  
01798 4.219e-7, 3.885e-7, 3.583e-7, 3.314e-7, 3.071e-7, 2.852e-7,  
01799 2.654e-7, 2.474e-7, 2.311e-7, 2.162e-7, 2.026e-7, 1.902e-7,  
01800 1.788e-7, 1.683e-7, 1.587e-7, 1.497e-7, 1.415e-7, 1.338e-7,  
01801 1.266e-7, 1.2e-7, 1.138e-7, 1.08e-7, 1.027e-7, 9.764e-8,  
01802 9.296e-8, 8.862e-8, 8.458e-8, 8.087e-8, 7.744e-8, 7.429e-8,  
01803 7.145e-8, 6.893e-8, 6.664e-8, 6.468e-8, 6.322e-8, 6.162e-8,  
01804 6.07e-8, 5.992e-8, 5.913e-8, 5.841e-8, 5.796e-8, 5.757e-8,  
01805 5.746e-8, 5.731e-8, 5.679e-8, 5.577e-8, 5.671e-8, 5.656e-8,  
01806 5.594e-8, 5.593e-8, 5.602e-8, 5.62e-8, 5.693e-8, 5.725e-8,  
01807 5.858e-8, 6.037e-8, 6.249e-8, 6.535e-8, 6.899e-8, 7.356e-8,  
01808 7.918e-8, 8.618e-8, 9.385e-8, 1.039e-7, 1.158e-7, 1.29e-7,  
01809 1.437e-7, 1.65e-7, 1.871e-7, 2.121e-7, 2.427e-7, 2.773e-7,  
01810 3.247e-7, 3.677e-7, 4.037e-7, 4.776e-7, 5.101e-7, 6.214e-7,  
01811 6.936e-7, 7.581e-7, 8.486e-7, 9.355e-7, 9.942e-7, 1.063e-6,  
01812 1.123e-6, 1.191e-6, 1.215e-6, 1.247e-6, 1.26e-6, 1.271e-6,  
01813 1.284e-6, 1.317e-6, 1.323e-6, 1.349e-6, 1.353e-6, 1.362e-6,  
01814 1.344e-6, 1.329e-6, 1.336e-6, 1.327e-6, 1.325e-6, 1.359e-6,  
01815 1.374e-6, 1.415e-6, 1.462e-6, 1.526e-6, 1.619e-6, 1.735e-6,  
01816 1.863e-6, 2.034e-6, 2.265e-6, 2.482e-6, 2.756e-6, 3.103e-6,  
01817 3.466e-6, 3.832e-6, 4.378e-6, 4.913e-6, 5.651e-6, 6.311e-6,  
01818 7.169e-6, 8.057e-6, 9.253e-6, 1.047e-5, 1.212e-5, 1.36e-5,  
01819 1.569e-5, 1.776e-5, 2.02e-5, 2.281e-5, 2.683e-5, 2.994e-5,  
01820 3.488e-5, 3.896e-5, 4.499e-5, 5.175e-5, 6.035e-5, 6.34e-5,  
01821 7.281e-5, 7.923e-5, 8.348e-5, 9.631e-5, 1.044e-4, 1.102e-4,  
01822 1.176e-4, 1.244e-4, 1.283e-4, 1.326e-4, 1.4e-4, 1.395e-4,  
01823 1.387e-4, 1.363e-4, 1.314e-4, 1.241e-4, 1.228e-4, 1.148e-4,  
01824 1.086e-4, 1.018e-4, 8.89e-5, 8.316e-5, 7.292e-5, 6.452e-5,  
01825 5.625e-5, 5.045e-5, 4.38e-5, 3.762e-5, 3.29e-5, 2.836e-5,  
01826 2.485e-5, 2.168e-5, 1.895e-5, 1.659e-5, 1.453e-5, 1.282e-5,  
01827 1.132e-5, 1.001e-5, 8.836e-6, 7.804e-6, 6.922e-6, 6.116e-6,  
01828 5.429e-6, 4.824e-6, 4.278e-6, 3.788e-6, 3.371e-6, 2.985e-6,  
01829 2.649e-6, 2.357e-6, 2.09e-6, 1.858e-6, 1.647e-6, 1.462e-6,  
01830 1.299e-6, 1.155e-6, 1.028e-6, 9.142e-7, 8.132e-7, 7.246e-7,  
01831 6.451e-7, 5.764e-7, 5.151e-7, 4.603e-7, 4.121e-7, 3.694e-7,  
01832 3.318e-7, 2.985e-7, 2.69e-7, 2.428e-7, 2.197e-7, 1.992e-7,  
01833 1.81e-7, 1.649e-7, 1.506e-7, 1.378e-7, 1.265e-7, 1.163e-7,  
01834 1.073e-7, 9.918e-8, 9.191e-8, 8.538e-8, 7.949e-8, 7.419e-8,  
01835 6.94e-8, 6.508e-8, 6.114e-8, 5.761e-8, 5.437e-8, 5.146e-8,  
01836 4.89e-8, 4.636e-8, 4.406e-8, 4.201e-8, 4.015e-8, 3.84e-8,  
01837 3.661e-8, 3.51e-8, 3.377e-8, 3.242e-8, 3.13e-8, 3.015e-8,  
01838 2.918e-8, 2.83e-8, 2.758e-8, 2.707e-8, 2.656e-8, 2.619e-8,  
01839 2.609e-8, 2.615e-8, 2.63e-8, 2.675e-8, 2.745e-8, 2.842e-8,  
01840 2.966e-8, 3.125e-8, 3.318e-8, 3.565e-8, 3.85e-8, 4.191e-8,  
01841 4.59e-8, 5.059e-8, 5.607e-8, 6.239e-8, 6.958e-8, 7.796e-8,  
01842 8.773e-8, 9.88e-8, 1.114e-7, 1.258e-7, 1.422e-7, 1.61e-7,  
01843 1.822e-7, 2.06e-7, 2.337e-7, 2.645e-7, 2.996e-7, 3.393e-7,  
01844 3.843e-7, 4.363e-7, 4.935e-7, 5.607e-7, 6.363e-7, 7.242e-7,  
01845 8.23e-7, 9.411e-7, 1.071e-6, 1.232e-6, 1.402e-6, 1.6e-6, 1.82e-6,  
01846 2.128e-6, 2.386e-6, 2.781e-6, 3.242e-6, 3.653e-6, 4.323e-6,  
01847 4.747e-6, 5.321e-6, 5.919e-6, 6.681e-6, 7.101e-6, 7.983e-6,  
01848 8.342e-6, 8.741e-6, 9.431e-6, 9.952e-6, 1.026e-5, 1.055e-5,  
01849 1.095e-5, 1.095e-5, 1.087e-5, 1.056e-5, 1.026e-5, 9.715e-6,  
01850 9.252e-6, 8.452e-6, 7.958e-6, 7.268e-6, 6.295e-6, 6.003e-6, 5e-6,  
01851 4.591e-6, 3.983e-6, 3.479e-6, 3.058e-6, 2.667e-6, 2.293e-6,  
01852 1.995e-6, 1.747e-6, 1.517e-6, 1.335e-6, 1.165e-6, 1.028e-6,  
01853 9.007e-7, 7.956e-7, 7.015e-7, 6.192e-7, 5.491e-7, 4.859e-7,  
01854 4.297e-7, 3.799e-7, 3.38e-7, 3.002e-7, 2.659e-7, 2.366e-7,  
01855 2.103e-7, 1.861e-7, 1.655e-7, 1.469e-7, 1.309e-7, 1.162e-7,  
01856 1.032e-7, 9.198e-8, 8.181e-8, 7.294e-8, 6.516e-8, 5.787e-8,  
01857 5.163e-8, 4.612e-8, 4.119e-8, 3.695e-8, 3.308e-8, 2.976e-8,  
01858 2.67e-8, 2.407e-8, 2.171e-8, 1.965e-8, 1.78e-8, 1.617e-8,  
01859 1.47e-8, 1.341e-8, 1.227e-8, 1.125e-8, 1.033e-8, 9.524e-9,  
01860 8.797e-9, 8.162e-9, 7.565e-9, 7.04e-9, 6.56e-9, 6.129e-9,  
01861 5.733e-9, 5.376e-9, 5.043e-9, 4.75e-9, 4.466e-9, 4.211e-9,  
01862 3.977e-9, 3.759e-9, 3.558e-9, 3.373e-9, 3.201e-9, 3.043e-9,  
01863 2.895e-9, 2.76e-9, 2.635e-9, 2.518e-9, 2.411e-9, 2.314e-9,  
01864 2.23e-9, 2.151e-9, 2.087e-9, 2.035e-9, 1.988e-9, 1.946e-9,  
01865 1.927e-9, 1.916e-9, 1.916e-9, 1.933e-9, 1.966e-9, 2.018e-9,  
01866 2.09e-9, 2.182e-9, 2.299e-9, 2.442e-9, 2.623e-9, 2.832e-9,  
01867 3.079e-9, 3.368e-9, 3.714e-9, 4.104e-9, 4.567e-9, 5.091e-9,  
01868 5.701e-9, 6.398e-9, 7.194e-9, 8.127e-9, 9.141e-9, 1.035e-8,  
01869 1.177e-8, 1.338e-8, 1.508e-8, 1.711e-8, 1.955e-8, 2.216e-8,  
01870 2.534e-8, 2.871e-8, 3.291e-8, 3.711e-8, 4.285e-8, 4.868e-8,  
01871 5.509e-8, 6.276e-8, 7.262e-8, 8.252e-8, 9.4e-8, 1.064e-7,  
01872 1.247e-7, 1.411e-7, 1.626e-7, 1.827e-7, 2.044e-7, 2.284e-7,  
01873 2.452e-7, 2.854e-7, 3.026e-7, 3.278e-7, 3.474e-7, 3.693e-7,  
01874 3.93e-7, 4.104e-7, 4.22e-7, 4.439e-7, 4.545e-7, 4.778e-7,  
01875 4.812e-7, 5.018e-7, 4.899e-7, 5.075e-7, 5.073e-7, 5.171e-7,

```
01876 5.131e-7, 5.25e-7, 5.617e-7, 5.846e-7, 6.239e-7, 6.696e-7,
01877 7.398e-7, 8.073e-7, 9.15e-7, 1.009e-6, 1.116e-6, 1.264e-6,
01878 1.439e-6, 1.644e-6, 1.856e-6, 2.147e-6, 2.317e-6, 2.713e-6,
01879 2.882e-6, 2.99e-6, 3.489e-6, 3.581e-6, 4.033e-6, 4.26e-6,
01880 4.543e-6, 4.84e-6, 4.826e-6, 5.013e-6, 5.252e-6, 5.277e-6,
01881 5.306e-6, 5.236e-6, 5.123e-6, 5.171e-6, 4.843e-6, 4.615e-6,
01882 4.385e-6, 3.97e-6, 3.693e-6, 3.231e-6, 2.915e-6, 2.495e-6,
01883 2.144e-6, 1.91e-6, 1.639e-6, 1.417e-6, 1.226e-6, 1.065e-6,
01884 9.29e-7, 8.142e-7, 7.161e-7, 6.318e-7, 5.581e-7, 4.943e-7,
01885 4.376e-7, 3.884e-7, 3.449e-7, 3.06e-7, 2.712e-7, 2.412e-7,
01886 2.139e-7, 1.903e-7, 1.689e-7, 1.499e-7, 1.331e-7, 1.183e-7,
01887 1.05e-7, 9.362e-8, 8.306e-8, 7.403e-8, 6.578e-8, 5.853e-8,
01888 5.216e-8, 4.632e-8, 4.127e-8, 3.678e-8, 3.279e-8, 2.923e-8,
01889 2.612e-8, 2.339e-8, 2.094e-8, 1.877e-8, 1.686e-8, 1.516e-8,
01890 1.366e-8, 1.234e-8, 1.114e-8, 1.012e-8, 9.182e-9, 8.362e-9,
01891 7.634e-9, 6.981e-9, 6.406e-9, 5.888e-9, 5.428e-9, 5.021e-9,
01892 4.65e-9, 4.326e-9, 4.033e-9, 3.77e-9, 3.536e-9, 3.327e-9,
01893 3.141e-9, 2.974e-9, 2.825e-9, 2.697e-9, 2.584e-9, 2.488e-9,
01894 2.406e-9, 2.34e-9, 2.292e-9, 2.259e-9, 2.244e-9, 2.243e-9,
01895 2.272e-9, 2.31e-9, 2.378e-9, 2.454e-9, 2.618e-9, 2.672e-9,
01896 2.831e-9, 3.05e-9, 3.225e-9, 3.425e-9, 3.677e-9, 3.968e-9,
01897 4.221e-9, 4.639e-9, 4.96e-9, 5.359e-9, 5.649e-9, 6.23e-9,
01898 6.716e-9, 7.218e-9, 7.746e-9, 7.988e-9, 8.627e-9, 8.999e-9,
01899 9.442e-9, 9.82e-9, 1.015e-8, 1.06e-8, 1.079e-8, 1.109e-8,
01900 1.137e-8, 1.186e-8, 1.18e-8, 1.187e-8, 1.194e-8, 1.192e-8,
01901 1.224e-8, 1.245e-8, 1.246e-8, 1.318e-8, 1.377e-8, 1.471e-8,
01902 1.582e-8, 1.713e-8, 1.853e-8, 2.063e-8, 2.27e-8, 2.567e-8,
01903 2.891e-8, 3.264e-8, 3.744e-8, 4.286e-8, 4.915e-8, 5.623e-8,
01904 6.336e-8, 7.293e-8, 8.309e-8, 9.319e-8, 1.091e-7, 1.243e-7,
01905 1.348e-7, 1.449e-7, 1.62e-7, 1.846e-7, 1.937e-7, 2.04e-7,
01906 2.179e-7, 2.298e-7, 2.433e-7, 2.439e-7, 2.464e-7, 2.611e-7,
01907 2.617e-7, 2.582e-7, 2.453e-7, 2.401e-7, 2.349e-7, 2.203e-7,
01908 2.066e-7, 1.939e-7, 1.78e-7, 1.558e-7, 1.391e-7, 1.203e-7,
01909 1.048e-7, 9.464e-8, 8.306e-8, 7.239e-8, 6.317e-8, 5.52e-8,
01910 4.847e-8, 4.282e-8, 3.796e-8, 3.377e-8, 2.996e-8, 2.678e-8,
01911 2.4e-8, 2.134e-8, 1.904e-8, 1.705e-8, 1.523e-8, 1.35e-8,
01912 1.204e-8, 1.07e-8, 9.408e-9, 8.476e-9, 7.47e-9, 6.679e-9,
01913 5.929e-9, 5.267e-9, 4.711e-9, 4.172e-9, 3.761e-9, 3.288e-9,
01914 2.929e-9, 2.609e-9, 2.315e-9, 2.042e-9, 1.844e-9, 1.64e-9,
01915 1.47e-9, 1.31e-9, 1.176e-9, 1.049e-9, 9.377e-10, 8.462e-10,
01916 7.616e-10, 6.854e-10, 6.191e-10, 5.596e-10, 5.078e-10, 4.611e-10,
01917 4.197e-10, 3.83e-10, 3.505e-10, 3.215e-10, 2.956e-10, 2.726e-10,
01918 2.521e-10, 2.338e-10, 2.173e-10, 2.026e-10, 1.895e-10, 1.777e-10,
01919 1.672e-10, 1.579e-10, 1.496e-10, 1.423e-10, 1.358e-10, 1.302e-10,
01920 1.254e-10, 1.216e-10, 1.187e-10, 1.163e-10, 1.147e-10, 1.145e-10,
01921 1.15e-10, 1.17e-10, 1.192e-10, 1.25e-10, 1.298e-10, 1.345e-10,
01922 1.405e-10, 1.538e-10, 1.648e-10, 1.721e-10, 1.872e-10, 1.968e-10,
01923 2.089e-10, 2.172e-10, 2.317e-10, 2.389e-10, 2.503e-10, 2.585e-10,
01924 2.686e-10, 2.8e-10, 2.895e-10, 3.019e-10, 3.037e-10, 3.076e-10,
01925 3.146e-10, 3.198e-10, 3.332e-10, 3.397e-10, 3.54e-10, 3.667e-10,
01926 3.895e-10, 4.071e-10, 4.565e-10, 4.983e-10, 5.439e-10, 5.968e-10,
01927 6.676e-10, 7.456e-10, 8.405e-10, 9.478e-10, 1.064e-9, 1.218e-9,
01928 1.386e-9, 1.581e-9, 1.787e-9, 2.032e-9, 2.347e-9, 2.677e-9,
01929 3.008e-9, 3.544e-9, 4.056e-9, 4.687e-9, 5.331e-9, 6.227e-9,
01930 6.854e-9, 8.139e-9, 8.945e-9, 9.865e-9, 1.125e-8, 1.178e-8,
01931 1.364e-8, 1.436e-8, 1.54e-8, 1.672e-8, 1.793e-8, 1.906e-8,
01932 2.036e-8, 2.144e-8, 2.292e-8, 2.371e-8, 2.493e-8, 2.606e-8,
01933 2.706e-8, 2.866e-8, 3.036e-8, 3.136e-8, 3.405e-8, 3.665e-8,
01934 3.837e-8, 4.229e-8, 4.748e-8, 5.32e-8, 5.763e-8, 6.677e-8,
01935 7.216e-8, 7.716e-8, 8.958e-8, 9.419e-8, 1.036e-7, 1.108e-7,
01936 1.189e-7, 1.246e-7, 1.348e-7, 1.31e-7, 1.361e-7, 1.364e-7,
01937 1.363e-7, 1.343e-7, 1.293e-7, 1.254e-7, 1.235e-7, 1.158e-7,
01938 1.107e-7, 9.961e-8, 9.011e-8, 7.91e-8, 6.916e-8, 6.338e-8,
01939 5.564e-8, 4.827e-8, 4.198e-8, 3.695e-8, 3.276e-8, 2.929e-8,
01940 2.633e-8, 2.391e-8, 2.192e-8, 2.021e-8, 1.89e-8, 1.772e-8,
01941 1.667e-8, 1.603e-8, 1.547e-8, 1.537e-8, 1.492e-8, 1.515e-8,
01942 1.479e-8, 1.45e-8, 1.513e-8, 1.495e-8, 1.529e-8, 1.565e-8,
01943 1.564e-8, 1.553e-8, 1.569e-8, 1.584e-8, 1.57e-8, 1.538e-8,
01944 1.513e-8, 1.472e-8, 1.425e-8, 1.349e-8, 1.328e-8, 1.249e-8,
01945 1.17e-8, 1.077e-8, 9.514e-9, 8.614e-9, 7.46e-9, 6.621e-9,
01946 5.775e-9, 5.006e-9, 4.308e-9, 3.747e-9, 3.24e-9, 2.84e-9,
01947 2.481e-9, 2.184e-9, 1.923e-9, 1.71e-9, 1.504e-9, 1.334e-9,
01948 1.187e-9, 1.053e-9, 9.367e-10, 8.306e-10, 7.419e-10, 6.63e-10,
01949 5.918e-10, 5.277e-10, 4.717e-10, 4.222e-10, 3.783e-10, 3.39e-10,
01950 3.036e-10, 2.729e-10, 2.455e-10, 2.211e-10, 1.995e-10, 1.804e-10,
01951 1.635e-10, 1.485e-10, 1.355e-10, 1.24e-10, 1.139e-10, 1.051e-10,
01952 9.757e-11, 9.114e-11, 8.577e-11, 8.139e-11, 7.792e-11, 7.52e-11,
01953 7.39e-11, 7.311e-11, 7.277e-11, 7.482e-11, 7.698e-11, 8.162e-11,
01954 8.517e-11, 8.968e-11, 9.905e-11, 1.075e-10, 1.187e-10, 1.291e-10,
01955 1.426e-10, 1.573e-10, 1.734e-10, 1.905e-10, 2.097e-10, 2.28e-10,
01956 2.473e-10, 2.718e-10, 2.922e-10, 3.128e-10, 3.361e-10, 3.641e-10,
01957 3.91e-10, 4.196e-10, 4.501e-10, 4.932e-10, 5.258e-10, 5.755e-10,
01958 6.253e-10, 6.664e-10, 7.344e-10, 7.985e-10, 8.877e-10, 1.005e-9,
01959 1.118e-9, 1.251e-9, 1.428e-9, 1.61e-9, 1.888e-9, 2.077e-9,
01960 2.331e-9, 2.751e-9, 3.061e-9, 3.522e-9, 3.805e-9, 4.181e-9,
01961 4.575e-9, 5.167e-9, 5.634e-9, 6.007e-9, 6.501e-9, 6.829e-9,
01962 7.211e-9, 7.262e-9, 7.696e-9, 7.832e-9, 7.799e-9, 7.651e-9,
```

01963 7.304e-9, 7.15e-9, 6.977e-9, 6.603e-9, 6.209e-9, 5.69e-9,  
01964 5.432e-9, 4.764e-9, 4.189e-9, 3.64e-9, 3.203e-9, 2.848e-9,  
01965 2.51e-9, 2.194e-9, 1.946e-9, 1.75e-9, 1.567e-9, 1.426e-9,  
01966 1.302e-9, 1.197e-9, 1.109e-9, 1.035e-9, 9.719e-10, 9.207e-10,  
01967 8.957e-10, 8.578e-10, 8.262e-10, 8.117e-10, 7.987e-10, 7.875e-10,  
01968 7.741e-10, 7.762e-10, 7.537e-10, 7.424e-10, 7.474e-10, 7.294e-10,  
01969 7.216e-10, 7.233e-10, 7.075e-10, 6.892e-10, 6.618e-10, 6.314e-10,  
01970 6.208e-10, 5.689e-10, 5.55e-10, 4.984e-10, 4.6e-10, 4.078e-10,  
01971 3.879e-10, 3.459e-10, 2.982e-10, 2.626e-10, 2.329e-10, 1.988e-10,  
01972 1.735e-10, 1.487e-10, 1.297e-10, 1.133e-10, 9.943e-11, 8.736e-11,  
01973 7.726e-11, 6.836e-11, 6.053e-11, 5.384e-11, 4.789e-11, 4.267e-11,  
01974 3.804e-11, 3.398e-11, 3.034e-11, 2.71e-11, 2.425e-11, 2.173e-11,  
01975 1.95e-11, 1.752e-11, 1.574e-11, 1.418e-11, 1.278e-11, 1.154e-11,  
01976 1.044e-11, 9.463e-12, 8.602e-12, 7.841e-12, 7.171e-12, 6.584e-12,  
01977 6.073e-12, 5.631e-12, 5.254e-12, 4.937e-12, 4.679e-12, 4.476e-12,  
01978 4.328e-12, 4.233e-12, 4.194e-12, 4.211e-12, 4.286e-12, 4.424e-12,  
01979 4.628e-12, 4.906e-12, 5.262e-12, 5.708e-12, 6.254e-12, 6.914e-12,  
01980 7.714e-12, 8.677e-12, 9.747e-12, 1.101e-11, 1.256e-11, 1.409e-11,  
01981 1.597e-11, 1.807e-11, 2.034e-11, 2.316e-11, 2.622e-11, 2.962e-11,  
01982 3.369e-11, 3.819e-11, 4.329e-11, 4.932e-11, 5.589e-11, 6.364e-11,  
01983 7.284e-11, 8.236e-11, 9.447e-11, 1.078e-10, 1.229e-10, 1.417e-10,  
01984 1.614e-10, 1.843e-10, 2.107e-10, 2.406e-10, 2.728e-10, 3.195e-10,  
01985 3.595e-10, 4.153e-10, 4.736e-10, 5.41e-10, 6.088e-10, 6.769e-10,  
01986 7.691e-10, 8.545e-10, 9.621e-10, 1.047e-9, 1.161e-9, 1.296e-9,  
01987 1.424e-9, 1.576e-9, 1.739e-9, 1.893e-9, 2.08e-9, 2.336e-9,  
01988 2.604e-9, 2.76e-9, 3.001e-9, 3.365e-9, 3.55e-9, 3.895e-9,  
01989 4.183e-9, 4.614e-9, 4.846e-9, 5.068e-9, 5.427e-9, 5.541e-9,  
01990 5.864e-9, 5.997e-9, 5.997e-9, 6.061e-9, 5.944e-9, 5.855e-9,  
01991 5.661e-9, 5.523e-9, 5.374e-9, 4.94e-9, 4.688e-9, 4.17e-9,  
01992 3.913e-9, 3.423e-9, 2.997e-9, 2.598e-9, 2.253e-9, 1.946e-9,  
01993 1.71e-9, 1.507e-9, 1.336e-9, 1.19e-9, 1.068e-9, 9.623e-10,  
01994 8.772e-10, 8.007e-10, 7.42e-10, 6.884e-10, 6.483e-10, 6.162e-10,  
01995 5.922e-10, 5.688e-10, 5.654e-10, 5.637e-10, 5.701e-10, 5.781e-10,  
01996 5.874e-10, 6.268e-10, 6.357e-10, 6.525e-10, 7.137e-10, 7.441e-10,  
01997 8.024e-10, 8.485e-10, 9.143e-10, 9.536e-10, 9.717e-10, 1.018e-9,  
01998 1.042e-9, 1.054e-9, 1.092e-9, 1.079e-9, 1.064e-9, 1.043e-9,  
01999 1.02e-9, 9.687e-10, 9.273e-10, 9.208e-10, 9.068e-10, 7.687e-10,  
02000 7.385e-10, 6.595e-10, 5.87e-10, 5.144e-10, 4.417e-10, 3.804e-10,  
02001 3.301e-10, 2.866e-10, 2.509e-10, 2.202e-10, 1.947e-10, 1.719e-10,  
02002 1.525e-10, 1.361e-10, 1.21e-10, 1.084e-10, 9.8e-11, 8.801e-11,  
02003 7.954e-11, 7.124e-11, 6.335e-11, 5.76e-11, 5.132e-11, 4.601e-11,  
02004 4.096e-11, 3.657e-11, 3.25e-11, 2.909e-11, 2.587e-11, 2.297e-11,  
02005 2.05e-11, 1.828e-11, 1.632e-11, 1.462e-11, 1.314e-11, 1.185e-11,  
02006 1.073e-11, 9.76e-12, 8.922e-12, 8.206e-12, 7.602e-12, 7.1e-12,  
02007 6.694e-12, 6.378e-12, 6.149e-12, 6.004e-12, 5.941e-12, 5.962e-12,  
02008 6.069e-12, 6.265e-12, 6.551e-12, 6.935e-12, 7.457e-12, 8.074e-12,  
02009 8.811e-12, 9.852e-12, 1.086e-11, 1.207e-11, 1.361e-11, 1.553e-11,  
02010 1.737e-11, 1.93e-11, 2.175e-11, 2.41e-11, 2.706e-11, 3.023e-11,  
02011 3.313e-11, 3.657e-11, 4.118e-11, 4.569e-11, 5.025e-11, 5.66e-11,  
02012 6.231e-11, 6.881e-11, 7.996e-11, 8.526e-11, 9.694e-11, 1.106e-10,  
02013 1.222e-10, 1.355e-10, 1.525e-10, 1.775e-10, 1.924e-10, 2.181e-10,  
02014 2.379e-10, 2.662e-10, 2.907e-10, 3.154e-10, 3.366e-10, 3.579e-10,  
02015 3.858e-10, 4.046e-10, 4.196e-10, 4.166e-10, 4.457e-10, 4.466e-10,  
02016 4.404e-10, 4.337e-10, 4.15e-10, 4.083e-10, 3.91e-10, 3.723e-10,  
02017 3.514e-10, 3.303e-10, 2.847e-10, 2.546e-10, 2.23e-10, 1.994e-10,  
02018 1.733e-10, 1.488e-10, 1.297e-10, 1.144e-10, 1.004e-10, 8.741e-11,  
02019 7.928e-11, 7.034e-11, 6.323e-11, 5.754e-11, 5.25e-11, 4.85e-11,  
02020 4.502e-11, 4.286e-11, 4.028e-11, 3.899e-11, 3.824e-11, 3.761e-11,  
02021 3.804e-11, 3.839e-11, 3.845e-11, 4.244e-11, 4.382e-11, 4.582e-11,  
02022 4.847e-11, 5.209e-11, 5.384e-11, 5.887e-11, 6.371e-11, 6.737e-11,  
02023 7.168e-11, 7.415e-11, 7.827e-11, 8.037e-11, 8.12e-11, 8.071e-11,  
02024 8.008e-11, 7.851e-11, 7.544e-11, 7.377e-11, 7.173e-11, 6.801e-11,  
02025 6.267e-11, 5.727e-11, 5.288e-11, 4.853e-11, 4.082e-11, 3.645e-11,  
02026 3.136e-11, 2.672e-11, 2.304e-11, 1.986e-11, 1.725e-11, 1.503e-11,  
02027 1.315e-11, 1.153e-11, 1.014e-11, 8.942e-12, 7.901e-12, 6.993e-12,  
02028 6.199e-12, 5.502e-12, 4.89e-12, 4.351e-12, 3.878e-12, 3.461e-12,  
02029 3.094e-12, 2.771e-12, 2.488e-12, 2.241e-12, 2.025e-12, 1.838e-12,  
02030 1.677e-12, 1.541e-12, 1.427e-12, 1.335e-12, 1.262e-12, 1.209e-12,  
02031 1.176e-12, 1.161e-12, 1.165e-12, 1.189e-12, 1.234e-12, 1.3e-12,  
02032 1.389e-12, 1.503e-12, 1.644e-12, 1.814e-12, 2.017e-12, 2.255e-12,  
02033 2.534e-12, 2.588e-12, 3.231e-12, 3.661e-12, 4.153e-12, 4.717e-12,  
02034 5.36e-12, 6.094e-12, 6.93e-12, 7.882e-12, 8.966e-12, 1.02e-11,  
02035 1.162e-11, 1.324e-11, 1.51e-11, 1.72e-11, 1.965e-11, 2.237e-11,  
02036 2.56e-11, 2.927e-11, 3.371e-11, 3.842e-11, 4.429e-11, 5.139e-11,  
02037 5.798e-11, 6.697e-11, 7.626e-11, 8.647e-11, 1.022e-10, 1.136e-10,  
02038 1.3e-10, 1.481e-10, 1.672e-10, 1.871e-10, 2.126e-10, 2.357e-10,  
02039 2.583e-10, 2.997e-10, 3.289e-10, 3.702e-10, 4.012e-10, 4.319e-10,  
02040 4.527e-10, 5.001e-10, 5.448e-10, 5.611e-10, 5.76e-10, 5.965e-10,  
02041 6.079e-10, 6.207e-10, 6.276e-10, 6.222e-10, 6.137e-10, 6e-10,  
02042 5.814e-10, 5.393e-10, 5.35e-10, 4.947e-10, 4.629e-10, 4.117e-10,  
02043 3.712e-10, 3.372e-10, 2.923e-10, 2.55e-10, 2.232e-10, 1.929e-10,  
02044 1.679e-10, 1.46e-10, 1.289e-10, 1.13e-10, 9.953e-11, 8.763e-11,  
02045 7.76e-11, 6.9e-11, 6.16e-11, 5.525e-11, 4.958e-11, 4.489e-11,  
02046 4.072e-11, 3.728e-11, 3.438e-11, 3.205e-11, 3.006e-11, 2.848e-11,  
02047 2.766e-11, 2.688e-11, 2.664e-11, 2.67e-11, 2.696e-11, 2.786e-11,  
02048 2.861e-11, 3.009e-11, 3.178e-11, 3.389e-11, 3.587e-11, 3.819e-11,  
02049 4.054e-11, 4.417e-11, 4.703e-11, 5.137e-11, 5.46e-11, 6.055e-11,

```
02050 6.333e-11, 6.773e-11, 7.219e-11, 7.717e-11, 8.131e-11, 8.491e-11,
02051 8.574e-11, 9.01e-11, 9.017e-11, 8.999e-11, 8.959e-11, 8.838e-11,
02052 8.579e-11, 8.162e-11, 8.098e-11, 7.472e-11, 7.108e-11, 6.559e-11,
02053 5.994e-11, 5.172e-11, 4.424e-11, 3.951e-11, 3.34e-11, 2.902e-11,
02054 2.541e-11, 2.215e-11, 1.945e-11, 1.716e-11, 1.503e-11, 1.339e-11,
02055 1.185e-11, 1.05e-11, 9.336e-12, 8.307e-12, 7.312e-12, 6.55e-12,
02056 5.836e-12, 5.178e-12, 4.6e-12, 4.086e-12, 3.639e-12, 3.247e-12,
02057 2.904e-12, 2.604e-12, 2.341e-12, 2.112e-12, 1.914e-12, 1.744e-12,
02058 1.598e-12, 1.476e-12, 1.374e-12, 1.293e-12, 1.23e-12, 1.185e-12,
02059 1.158e-12, 1.147e-12, 1.154e-12, 1.177e-12, 1.219e-12, 1.28e-12,
02060 1.36e-12, 1.463e-12, 1.591e-12, 1.75e-12, 1.94e-12, 2.156e-12,
02061 2.43e-12, 2.748e-12, 3.052e-12, 3.533e-12, 3.967e-12, 4.471e-12,
02062 5.041e-12, 5.86e-12, 6.664e-12, 7.522e-12, 8.342e-12, 9.412e-12,
02063 1.072e-11, 1.213e-11, 1.343e-11, 1.496e-11, 1.664e-11, 1.822e-11,
02064 2.029e-11, 2.233e-11, 2.457e-11, 2.709e-11, 2.928e-11, 3.115e-11,
02065 3.356e-11, 3.592e-11, 3.818e-11, 3.936e-11, 4.061e-11, 4.149e-11,
02066 4.299e-11, 4.223e-11, 4.251e-11, 4.287e-11, 4.177e-11, 4.094e-11,
02067 3.942e-11, 3.772e-11, 3.614e-11, 3.394e-11, 3.222e-11, 2.791e-11,
02068 2.665e-11, 2.309e-11, 2.032e-11, 1.74e-11, 1.535e-11, 1.323e-11,
02069 1.151e-11, 9.803e-12, 8.65e-12, 7.54e-12, 6.619e-12, 5.832e-12,
02070 5.113e-12, 4.503e-12, 3.975e-12, 3.52e-12, 3.112e-12, 2.797e-12,
02071 2.5e-12, 2.24e-12, 2.013e-12, 1.819e-12, 1.653e-12, 1.513e-12,
02072 1.395e-12, 1.299e-12, 1.225e-12, 1.168e-12, 1.124e-12, 1.148e-12,
02073 1.107e-12, 1.128e-12, 1.169e-12, 1.233e-12, 1.307e-12, 1.359e-12,
02074 1.543e-12, 1.686e-12, 1.794e-12, 2.028e-12, 2.21e-12, 2.441e-12,
02075 2.653e-12, 2.828e-12, 3.093e-12, 3.28e-12, 3.551e-12, 3.677e-12,
02076 3.803e-12, 3.844e-12, 4.068e-12, 4.093e-12, 4.002e-12, 3.904e-12,
02077 3.624e-12, 3.633e-12, 3.622e-12, 3.443e-12, 3.184e-12, 2.934e-12,
02078 2.476e-12, 2.212e-12, 1.867e-12, 1.594e-12, 1.37e-12, 1.192e-12,
02079 1.045e-12, 9.211e-13, 8.17e-13, 7.29e-13, 6.55e-13, 5.929e-13,
02080 5.415e-13, 4.995e-13, 4.661e-13, 4.406e-13, 4.225e-13, 4.116e-13,
02081 4.075e-13, 4.102e-13, 4.198e-13, 4.365e-13, 4.606e-13, 4.925e-13,
02082 5.326e-13, 5.818e-13, 6.407e-13, 7.104e-13, 7.92e-13, 8.868e-13,
02083 9.964e-13, 1.123e-12, 1.268e-12, 1.434e-12, 1.626e-12, 1.848e-12,
02084 2.107e-12, 2.422e-12, 2.772e-12, 3.145e-12, 3.704e-12, 4.27e-12,
02085 4.721e-12, 5.361e-12, 6.083e-12, 7.095e-12, 7.968e-12, 9.228e-12,
02086 1.048e-11, 1.187e-11, 1.336e-11, 1.577e-11, 1.772e-11, 2.017e-11,
02087 2.25e-11, 2.63e-11, 2.911e-11, 3.356e-11, 3.82e-11, 4.173e-11,
02088 4.811e-11, 5.254e-11, 5.839e-11, 6.187e-11, 6.805e-11, 7.118e-11,
02089 7.369e-11, 7.664e-11, 7.794e-11, 7.947e-11, 8.036e-11, 7.954e-11,
02090 7.849e-11, 7.518e-11, 7.462e-11, 6.926e-11, 6.531e-11, 6.197e-11,
02091 5.421e-11, 4.777e-11, 4.111e-11, 3.679e-11, 3.166e-11, 2.786e-11,
02092 2.436e-11, 2.144e-11, 1.859e-11, 1.628e-11, 1.414e-11, 1.237e-11,
02093 1.093e-11, 9.558e-12
02094 };
02095
02096 static double h2o260[2001] = { .2752, .2732, .2749, .2676, .2667, .2545,
02097 .2497, .2327, .2218, .2036, .1825, .1694, .1497, .1353, .121,
02098 .1014, .09405, .07848, .07195, .06246, .05306, .04853, .04138,
02099 .03735, .03171, .02785, .02431, .02111, .01845, .0164, .01405,
02100 .01255, .01098, .009797, .008646, .007779, .006898, .006099,
02101 .005453, .004909, .004413, .003959, .003581, .003199, .002871,
02102 .002583, .00233, .002086, .001874, .001684, .001512, .001361,
02103 .001225, .0011, 9.89e-4, 8.916e-4, 8.039e-4, 7.256e-4, 6.545e-4,
02104 5.918e-4, 5.359e-4, 4.867e-4, 4.426e-4, 4.033e-4, 3.682e-4,
02105 3.366e-4, 3.085e-4, 2.833e-4, 2.605e-4, 2.403e-4, 2.221e-4,
02106 2.055e-4, 1.908e-4, 1.774e-4, 1.653e-4, 1.544e-4, 1.443e-4,
02107 1.351e-4, 1.267e-4, 1.19e-4, 1.119e-4, 1.053e-4, 9.922e-5,
02108 9.355e-5, 8.831e-5, 8.339e-5, 7.878e-5, 7.449e-5, 7.043e-5,
02109 6.664e-5, 6.307e-5, 5.969e-5, 5.654e-5, 5.357e-5, 5.075e-5,
02110 4.81e-5, 4.56e-5, 4.322e-5, 4.102e-5, 3.892e-5, 3.696e-5,
02111 3.511e-5, 3.339e-5, 3.177e-5, 3.026e-5, 2.886e-5, 2.756e-5,
02112 2.636e-5, 2.527e-5, 2.427e-5, 2.337e-5, 2.257e-5, 2.185e-5,
02113 2.127e-5, 2.08e-5, 2.041e-5, 2.013e-5, 2e-5, 1.997e-5, 2.009e-5,
02114 2.031e-5, 2.068e-5, 2.124e-5, 2.189e-5, 2.267e-5, 2.364e-5,
02115 2.463e-5, 2.618e-5, 2.774e-5, 2.937e-5, 3.144e-5, 3.359e-5,
02116 3.695e-5, 4.002e-5, 4.374e-5, 4.947e-5, 5.431e-5, 6.281e-5,
02117 7.169e-5, 8.157e-5, 9.728e-5, 1.079e-4, 1.337e-4, 1.442e-4,
02118 1.683e-4, 1.879e-4, 2.223e-4, 2.425e-4, 2.838e-4, 3.143e-4,
02119 3.527e-4, 4.012e-4, 4.237e-4, 4.747e-4, 5.057e-4, 5.409e-4,
02120 5.734e-4, 5.944e-4, 6.077e-4, 6.175e-4, 6.238e-4, 6.226e-4,
02121 6.248e-4, 6.192e-4, 6.098e-4, 5.818e-4, 5.709e-4, 5.465e-4,
02122 5.043e-4, 4.699e-4, 4.294e-4, 3.984e-4, 3.672e-4, 3.152e-4,
02123 2.883e-4, 2.503e-4, 2.211e-4, 1.92e-4, 1.714e-4, 1.485e-4,
02124 1.358e-4, 1.156e-4, 1.021e-4, 8.887e-5, 7.842e-5, 7.12e-5,
02125 6.186e-5, 5.73e-5, 4.792e-5, 4.364e-5, 3.72e-5, 3.28e-5,
02126 2.946e-5, 2.591e-5, 2.261e-5, 2.048e-5, 1.813e-5, 1.63e-5,
02127 1.447e-5, 1.282e-5, 1.167e-5, 1.041e-5, 9.449e-6, 8.51e-6,
02128 7.596e-6, 6.961e-6, 6.272e-6, 5.728e-6, 5.198e-6, 4.667e-6,
02129 4.288e-6, 3.897e-6, 3.551e-6, 3.235e-6, 2.952e-6, 2.688e-6,
02130 2.449e-6, 2.241e-6, 2.05e-6, 1.879e-6, 1.722e-6, 1.582e-6,
02131 1.456e-6, 1.339e-6, 1.236e-6, 1.144e-6, 1.06e-6, 9.83e-7,
02132 9.149e-7, 8.535e-7, 7.973e-7, 7.466e-7, 6.999e-7, 6.574e-7,
02133 6.18e-7, 5.821e-7, 5.487e-7, 5.18e-7, 4.896e-7, 4.631e-7,
02134 4.386e-7, 4.16e-7, 3.945e-7, 3.748e-7, 3.562e-7, 3.385e-7,
02135 3.222e-7, 3.068e-7, 2.922e-7, 2.788e-7, 2.659e-7, 2.539e-7,
02136 2.425e-7, 2.318e-7, 2.219e-7, 2.127e-7, 2.039e-7, 1.958e-7,
```



02137 1.885e-7, 1.818e-7, 1.758e-7, 1.711e-7, 1.662e-7, 1.63e-7,  
02138 1.605e-7, 1.58e-7, 1.559e-7, 1.545e-7, 1.532e-7, 1.522e-7,  
02139 1.51e-7, 1.495e-7, 1.465e-7, 1.483e-7, 1.469e-7, 1.448e-7,  
02140 1.444e-7, 1.436e-7, 1.426e-7, 1.431e-7, 1.425e-7, 1.445e-7,  
02141 1.477e-7, 1.515e-7, 1.567e-7, 1.634e-7, 1.712e-7, 1.802e-7,  
02142 1.914e-7, 2.024e-7, 2.159e-7, 2.295e-7, 2.461e-7, 2.621e-7,  
02143 2.868e-7, 3.102e-7, 3.394e-7, 3.784e-7, 4.223e-7, 4.864e-7,  
02144 5.501e-7, 6.039e-7, 7.193e-7, 7.728e-7, 9.514e-7, 1.073e-6,  
02145 1.18e-6, 1.333e-6, 1.472e-6, 1.566e-6, 1.677e-6, 1.784e-6,  
02146 1.904e-6, 1.953e-6, 2.02e-6, 2.074e-6, 2.128e-6, 2.162e-6,  
02147 2.219e-6, 2.221e-6, 2.249e-6, 2.239e-6, 2.235e-6, 2.185e-6,  
02148 2.141e-6, 2.124e-6, 2.09e-6, 2.068e-6, 2.1e-6, 2.104e-6,  
02149 2.142e-6, 2.181e-6, 2.257e-6, 2.362e-6, 2.5e-6, 2.664e-6,  
02150 2.884e-6, 3.189e-6, 3.48e-6, 3.847e-6, 4.313e-6, 4.79e-6,  
02151 5.25e-6, 5.989e-6, 6.692e-6, 7.668e-6, 8.52e-6, 9.606e-6,  
02152 1.073e-5, 1.225e-5, 1.377e-5, 1.582e-5, 1.761e-5, 2.029e-5,  
02153 2.284e-5, 2.602e-5, 2.94e-5, 3.483e-5, 3.928e-5, 4.618e-5,  
02154 5.24e-5, 6.132e-5, 7.183e-5, 8.521e-5, 9.111e-5, 1.07e-4,  
02155 1.184e-4, 1.264e-4, 1.475e-4, 1.612e-4, 1.704e-4, 1.818e-4,  
02156 1.924e-4, 1.994e-4, 2.061e-4, 2.18e-4, 2.187e-4, 2.2e-4,  
02157 2.196e-4, 2.131e-4, 2.015e-4, 1.988e-4, 1.847e-4, 1.729e-4,  
02158 1.597e-4, 1.373e-4, 1.262e-4, 1.087e-4, 9.439e-5, 8.061e-5,  
02159 7.093e-5, 6.049e-5, 5.12e-5, 4.435e-5, 3.817e-5, 3.34e-5,  
02160 2.927e-5, 2.573e-5, 2.291e-5, 2.04e-5, 1.827e-5, 1.636e-5,  
02161 1.463e-5, 1.309e-5, 1.17e-5, 1.047e-5, 9.315e-6, 8.328e-6,  
02162 7.458e-6, 6.665e-6, 5.94e-6, 5.316e-6, 4.752e-6, 4.252e-6,  
02163 3.825e-6, 3.421e-6, 3.064e-6, 2.746e-6, 2.465e-6, 2.216e-6,  
02164 1.99e-6, 1.79e-6, 1.609e-6, 1.449e-6, 1.306e-6, 1.177e-6,  
02165 1.063e-6, 9.607e-7, 8.672e-7, 7.855e-7, 7.118e-7, 6.46e-7,  
02166 5.871e-7, 5.34e-7, 4.868e-7, 4.447e-7, 4.068e-7, 3.729e-7,  
02167 3.423e-7, 3.151e-7, 2.905e-7, 2.686e-7, 2.484e-7, 2.306e-7,  
02168 2.142e-7, 1.995e-7, 1.86e-7, 1.738e-7, 1.626e-7, 1.522e-7,  
02169 1.427e-7, 1.338e-7, 1.258e-7, 1.183e-7, 1.116e-7, 1.056e-7,  
02170 9.972e-8, 9.46e-8, 9.007e-8, 8.592e-8, 8.195e-8, 7.816e-8,  
02171 7.483e-8, 7.193e-8, 6.892e-8, 6.642e-8, 6.386e-8, 6.154e-8,  
02172 5.949e-8, 5.764e-8, 5.622e-8, 5.479e-8, 5.364e-8, 5.301e-8,  
02173 5.267e-8, 5.263e-8, 5.313e-8, 5.41e-8, 5.55e-8, 5.745e-8,  
02174 6.003e-8, 6.311e-8, 6.713e-8, 7.173e-8, 7.724e-8, 8.368e-8,  
02175 9.121e-8, 9.986e-8, 1.097e-7, 1.209e-7, 1.338e-7, 1.486e-7,  
02176 1.651e-7, 1.837e-7, 2.048e-7, 2.289e-7, 2.557e-7, 2.857e-7,  
02177 3.195e-7, 3.587e-7, 4.015e-7, 4.497e-7, 5.049e-7, 5.665e-7,  
02178 6.366e-7, 7.121e-7, 7.996e-7, 8.946e-7, 1.002e-6, 1.117e-6,  
02179 1.262e-6, 1.416e-6, 1.611e-6, 1.807e-6, 2.056e-6, 2.351e-6,  
02180 2.769e-6, 3.138e-6, 3.699e-6, 4.386e-6, 5.041e-6, 6.074e-6,  
02181 6.812e-6, 7.79e-6, 8.855e-6, 1.014e-5, 1.095e-5, 1.245e-5,  
02182 1.316e-5, 1.39e-5, 1.504e-5, 1.583e-5, 1.617e-5, 1.652e-5,  
02183 1.713e-5, 1.724e-5, 1.715e-5, 1.668e-5, 1.629e-5, 1.552e-5,  
02184 1.478e-5, 1.34e-5, 1.245e-5, 1.121e-5, 9.575e-6, 8.956e-6,  
02185 7.345e-6, 6.597e-6, 5.612e-6, 4.818e-6, 4.165e-6, 3.579e-6,  
02186 3.041e-6, 2.623e-6, 2.29e-6, 1.984e-6, 1.748e-6, 1.534e-6,  
02187 1.369e-6, 1.219e-6, 1.092e-6, 9.8e-7, 8.762e-7, 7.896e-7,  
02188 7.104e-7, 6.364e-7, 5.691e-7, 5.107e-7, 4.575e-7, 4.09e-7,  
02189 3.667e-7, 3.287e-7, 2.931e-7, 2.633e-7, 2.356e-7, 2.111e-7,  
02190 1.895e-7, 1.697e-7, 1.525e-7, 1.369e-7, 1.233e-7, 1.114e-7,  
02191 9.988e-8, 9.004e-8, 8.149e-8, 7.352e-8, 6.662e-8, 6.03e-8,  
02192 5.479e-8, 4.974e-8, 4.532e-8, 4.129e-8, 3.781e-8, 3.462e-8,  
02193 3.176e-8, 2.919e-8, 2.687e-8, 2.481e-8, 2.292e-8, 2.119e-8,  
02194 1.967e-8, 1.828e-8, 1.706e-8, 1.589e-8, 1.487e-8, 1.393e-8,  
02195 1.307e-8, 1.228e-8, 1.156e-8, 1.089e-8, 1.028e-8, 9.696e-9,  
02196 9.159e-9, 8.658e-9, 8.187e-9, 7.746e-9, 7.34e-9, 6.953e-9,  
02197 6.594e-9, 6.259e-9, 5.948e-9, 5.66e-9, 5.386e-9, 5.135e-9,  
02198 4.903e-9, 4.703e-9, 4.515e-9, 4.362e-9, 4.233e-9, 4.117e-9,  
02199 4.017e-9, 3.962e-9, 3.924e-9, 3.905e-9, 3.922e-9, 3.967e-9,  
02200 4.046e-9, 4.165e-9, 4.32e-9, 4.522e-9, 4.769e-9, 5.083e-9,  
02201 5.443e-9, 5.872e-9, 6.366e-9, 6.949e-9, 7.601e-9, 8.371e-9,  
02202 9.22e-9, 1.02e-8, 1.129e-8, 1.251e-8, 1.393e-8, 1.542e-8,  
02203 1.72e-8, 1.926e-8, 2.152e-8, 2.392e-8, 2.678e-8, 3.028e-8,  
02204 3.39e-8, 3.836e-8, 4.309e-8, 4.9e-8, 5.481e-8, 6.252e-8,  
02205 7.039e-8, 7.883e-8, 8.849e-8, 1.012e-7, 1.142e-7, 1.3e-7,  
02206 1.475e-7, 1.732e-7, 1.978e-7, 2.304e-7, 2.631e-7, 2.988e-7,  
02207 3.392e-7, 3.69e-7, 4.355e-7, 4.672e-7, 5.11e-7, 5.461e-7,  
02208 5.828e-7, 6.233e-7, 6.509e-7, 6.672e-7, 6.969e-7, 7.104e-7,  
02209 7.439e-7, 7.463e-7, 7.708e-7, 7.466e-7, 7.668e-7, 7.549e-7,  
02210 7.586e-7, 7.384e-7, 7.439e-7, 7.785e-7, 7.915e-7, 8.31e-7,  
02211 8.745e-7, 9.558e-7, 1.038e-6, 1.173e-6, 1.304e-6, 1.452e-6,  
02212 1.671e-6, 1.931e-6, 2.239e-6, 2.578e-6, 3.032e-6, 3.334e-6,  
02213 3.98e-6, 4.3e-6, 4.518e-6, 5.321e-6, 5.508e-6, 6.211e-6, 6.59e-6,  
02214 7.046e-6, 7.555e-6, 7.558e-6, 7.875e-6, 8.319e-6, 8.433e-6,  
02215 8.59e-6, 8.503e-6, 8.304e-6, 8.336e-6, 7.739e-6, 7.301e-6,  
02216 6.827e-6, 6.078e-6, 5.551e-6, 4.762e-6, 4.224e-6, 3.538e-6,  
02217 2.984e-6, 2.619e-6, 2.227e-6, 1.923e-6, 1.669e-6, 1.462e-6,  
02218 1.294e-6, 1.155e-6, 1.033e-6, 9.231e-7, 8.238e-7, 7.36e-7,  
02219 6.564e-7, 5.869e-7, 5.236e-7, 4.673e-7, 4.174e-7, 3.736e-7,  
02220 3.33e-7, 2.976e-7, 2.657e-7, 2.367e-7, 2.106e-7, 1.877e-7,  
02221 1.671e-7, 1.494e-7, 1.332e-7, 1.192e-7, 1.065e-7, 9.558e-8,  
02222 8.586e-8, 7.717e-8, 6.958e-8, 6.278e-8, 5.666e-8, 5.121e-8,  
02223 4.647e-8, 4.213e-8, 3.815e-8, 3.459e-8, 3.146e-8, 2.862e-8,

```
02224 2.604e-8, 2.375e-8, 2.162e-8, 1.981e-8, 1.817e-8, 1.67e-8,
02225 1.537e-8, 1.417e-8, 1.31e-8, 1.215e-8, 1.128e-8, 1.05e-8,
02226 9.793e-9, 9.158e-9, 8.586e-9, 8.068e-9, 7.595e-9, 7.166e-9,
02227 6.778e-9, 6.427e-9, 6.108e-9, 5.826e-9, 5.571e-9, 5.347e-9,
02228 5.144e-9, 4.968e-9, 4.822e-9, 4.692e-9, 4.589e-9, 4.506e-9,
02229 4.467e-9, 4.44e-9, 4.466e-9, 4.515e-9, 4.718e-9, 4.729e-9,
02230 4.937e-9, 5.249e-9, 5.466e-9, 5.713e-9, 6.03e-9, 6.436e-9,
02231 6.741e-9, 7.33e-9, 7.787e-9, 8.414e-9, 8.908e-9, 9.868e-9,
02232 1.069e-8, 1.158e-8, 1.253e-8, 1.3e-8, 1.409e-8, 1.47e-8,
02233 1.548e-8, 1.612e-8, 1.666e-8, 1.736e-8, 1.763e-8, 1.812e-8,
02234 1.852e-8, 1.923e-8, 1.897e-8, 1.893e-8, 1.888e-8, 1.868e-8,
02235 1.895e-8, 1.899e-8, 1.876e-8, 1.96e-8, 2.02e-8, 2.121e-8,
02236 2.239e-8, 2.379e-8, 2.526e-8, 2.766e-8, 2.994e-8, 3.332e-8,
02237 3.703e-8, 4.158e-8, 4.774e-8, 5.499e-8, 6.355e-8, 7.349e-8,
02238 8.414e-8, 9.846e-8, 1.143e-7, 1.307e-7, 1.562e-7, 1.817e-7,
02239 2.011e-7, 2.192e-7, 2.485e-7, 2.867e-7, 3.035e-7, 3.223e-7,
02240 3.443e-7, 3.617e-7, 3.793e-7, 3.793e-7, 3.839e-7, 4.081e-7,
02241 4.117e-7, 4.085e-7, 3.92e-7, 3.851e-7, 3.754e-7, 3.49e-7,
02242 3.229e-7, 2.978e-7, 2.691e-7, 2.312e-7, 2.029e-7, 1.721e-7,
02243 1.472e-7, 1.308e-7, 1.132e-7, 9.736e-8, 8.458e-8, 7.402e-8,
02244 6.534e-8, 5.811e-8, 5.235e-8, 4.762e-8, 4.293e-8, 3.896e-8,
02245 3.526e-8, 3.165e-8, 2.833e-8, 2.551e-8, 2.288e-8, 2.036e-8,
02246 1.82e-8, 1.626e-8, 1.438e-8, 1.299e-8, 1.149e-8, 1.03e-8,
02247 9.148e-9, 8.122e-9, 7.264e-9, 6.425e-9, 5.777e-9, 5.06e-9,
02248 4.502e-9, 4.013e-9, 3.567e-9, 3.145e-9, 2.864e-9, 2.553e-9,
02249 2.311e-9, 2.087e-9, 1.886e-9, 1.716e-9, 1.556e-9, 1.432e-9,
02250 1.311e-9, 1.202e-9, 1.104e-9, 1.013e-9, 9.293e-10, 8.493e-10,
02251 7.79e-10, 7.185e-10, 6.642e-10, 6.141e-10, 5.684e-10, 5.346e-10,
02252 5.032e-10, 4.725e-10, 4.439e-10, 4.176e-10, 3.93e-10, 3.714e-10,
02253 3.515e-10, 3.332e-10, 3.167e-10, 3.02e-10, 2.887e-10, 2.769e-10,
02254 2.665e-10, 2.578e-10, 2.503e-10, 2.436e-10, 2.377e-10, 2.342e-10,
02255 2.305e-10, 2.296e-10, 2.278e-10, 2.321e-10, 2.355e-10, 2.402e-10,
02256 2.478e-10, 2.67e-10, 2.848e-10, 2.982e-10, 3.263e-10, 3.438e-10,
02257 3.649e-10, 3.829e-10, 4.115e-10, 4.264e-10, 4.473e-10, 4.63e-10,
02258 4.808e-10, 4.995e-10, 5.142e-10, 5.313e-10, 5.318e-10, 5.358e-10,
02259 5.452e-10, 5.507e-10, 5.698e-10, 5.782e-10, 5.983e-10, 6.164e-10,
02260 6.532e-10, 6.811e-10, 7.624e-10, 8.302e-10, 9.067e-10, 9.937e-10,
02261 1.104e-9, 1.221e-9, 1.361e-9, 1.516e-9, 1.675e-9, 1.883e-9,
02262 2.101e-9, 2.349e-9, 2.614e-9, 2.92e-9, 3.305e-9, 3.724e-9,
02263 4.142e-9, 4.887e-9, 5.614e-9, 6.506e-9, 7.463e-9, 8.817e-9,
02264 9.849e-9, 1.187e-8, 1.321e-8, 1.474e-8, 1.698e-8, 1.794e-8,
02265 2.09e-8, 2.211e-8, 2.362e-8, 2.556e-8, 2.729e-8, 2.88e-8,
02266 3.046e-8, 3.167e-8, 3.367e-8, 3.457e-8, 3.59e-8, 3.711e-8,
02267 3.826e-8, 4.001e-8, 4.211e-8, 4.315e-8, 4.661e-8, 5.01e-8,
02268 5.249e-8, 5.84e-8, 6.628e-8, 7.512e-8, 8.253e-8, 9.722e-8,
02269 1.067e-7, 1.153e-7, 1.347e-7, 1.428e-7, 1.577e-7, 1.694e-7,
02270 1.833e-7, 1.938e-7, 2.108e-7, 2.059e-7, 2.157e-7, 2.185e-7,
02271 2.208e-7, 2.182e-7, 2.093e-7, 2.014e-7, 1.962e-7, 1.819e-7,
02272 1.713e-7, 1.51e-7, 1.34e-7, 1.154e-7, 9.89e-8, 8.88e-8, 7.673e-8,
02273 6.599e-8, 5.73e-8, 5.081e-8, 4.567e-8, 4.147e-8, 3.773e-8,
02274 3.46e-8, 3.194e-8, 2.953e-8, 2.759e-8, 2.594e-8, 2.442e-8,
02275 2.355e-8, 2.283e-8, 2.279e-8, 2.231e-8, 2.279e-8, 2.239e-8,
02276 2.21e-8, 2.309e-8, 2.293e-8, 2.352e-8, 2.415e-8, 2.43e-8,
02277 2.426e-8, 2.465e-8, 2.5e-8, 2.496e-8, 2.465e-8, 2.445e-8,
02278 2.383e-8, 2.299e-8, 2.165e-8, 2.113e-8, 1.968e-8, 1.819e-8,
02279 1.644e-8, 1.427e-8, 1.27e-8, 1.082e-8, 9.428e-9, 8.091e-9,
02280 6.958e-9, 5.988e-9, 5.246e-9, 4.601e-9, 4.098e-9, 3.664e-9,
02281 3.287e-9, 2.942e-9, 2.656e-9, 2.364e-9, 2.118e-9, 1.903e-9,
02282 1.703e-9, 1.525e-9, 1.365e-9, 1.229e-9, 1.107e-9, 9.96e-10,
02283 8.945e-10, 8.08e-10, 7.308e-10, 6.616e-10, 5.994e-10, 5.422e-10,
02284 4.929e-10, 4.478e-10, 4.07e-10, 3.707e-10, 3.379e-10, 3.087e-10,
02285 2.823e-10, 2.592e-10, 2.385e-10, 2.201e-10, 2.038e-10, 1.897e-10,
02286 1.774e-10, 1.667e-10, 1.577e-10, 1.502e-10, 1.437e-10, 1.394e-10,
02287 1.358e-10, 1.324e-10, 1.329e-10, 1.324e-10, 1.36e-10, 1.39e-10,
02288 1.424e-10, 1.544e-10, 1.651e-10, 1.817e-10, 1.984e-10, 2.195e-10,
02289 2.438e-10, 2.7e-10, 2.991e-10, 3.322e-10, 3.632e-10, 3.957e-10,
02290 4.36e-10, 4.701e-10, 5.03e-10, 5.381e-10, 5.793e-10, 6.19e-10,
02291 6.596e-10, 7.004e-10, 7.561e-10, 7.934e-10, 8.552e-10, 9.142e-10,
02292 9.57e-10, 1.027e-9, 1.097e-9, 1.193e-9, 1.334e-9, 1.47e-9,
02293 1.636e-9, 1.871e-9, 2.122e-9, 2.519e-9, 2.806e-9, 3.203e-9,
02294 3.846e-9, 4.362e-9, 5.114e-9, 5.643e-9, 6.305e-9, 6.981e-9,
02295 7.983e-9, 8.783e-9, 9.419e-9, 1.017e-8, 1.063e-8, 1.121e-8,
02296 1.13e-8, 1.201e-8, 1.225e-8, 1.232e-8, 1.223e-8, 1.177e-8,
02297 1.151e-8, 1.116e-8, 1.047e-8, 9.698e-9, 8.734e-9, 8.202e-9,
02298 7.041e-9, 6.074e-9, 5.172e-9, 4.468e-9, 3.913e-9, 3.414e-9,
02299 2.975e-9, 2.65e-9, 2.406e-9, 2.173e-9, 2.009e-9, 1.861e-9,
02300 1.727e-9, 1.612e-9, 1.514e-9, 1.43e-9, 1.362e-9, 1.333e-9,
02301 1.288e-9, 1.249e-9, 1.238e-9, 1.228e-9, 1.217e-9, 1.202e-9,
02302 1.209e-9, 1.177e-9, 1.157e-9, 1.165e-9, 1.142e-9, 1.131e-9,
02303 1.138e-9, 1.117e-9, 1.1e-9, 1.069e-9, 1.023e-9, 1.005e-9,
02304 9.159e-10, 8.863e-10, 7.865e-10, 7.153e-10, 6.247e-10, 5.846e-10,
02305 5.133e-10, 4.36e-10, 3.789e-10, 3.335e-10, 2.833e-10, 2.483e-10,
02306 2.155e-10, 1.918e-10, 1.709e-10, 1.529e-10, 1.374e-10, 1.235e-10,
02307 1.108e-10, 9.933e-11, 8.932e-11, 8.022e-11, 7.224e-11, 6.52e-11,
02308 5.896e-11, 5.328e-11, 4.813e-11, 4.365e-11, 3.961e-11, 3.594e-11,
02309 3.266e-11, 2.967e-11, 2.701e-11, 2.464e-11, 2.248e-11, 2.054e-11,
02310 1.878e-11, 1.721e-11, 1.579e-11, 1.453e-11, 1.341e-11, 1.241e-11,
```

02311 1.154e-11, 1.078e-11, 1.014e-11, 9.601e-12, 9.167e-12, 8.838e-12,  
02312 8.614e-12, 8.493e-12, 8.481e-12, 8.581e-12, 8.795e-12, 9.131e-12,  
02313 9.601e-12, 1.021e-11, 1.097e-11, 1.191e-11, 1.303e-11, 1.439e-11,  
02314 1.601e-11, 1.778e-11, 1.984e-11, 2.234e-11, 2.474e-11, 2.766e-11,  
02315 3.085e-11, 3.415e-11, 3.821e-11, 4.261e-11, 4.748e-11, 5.323e-11,  
02316 5.935e-11, 6.619e-11, 7.418e-11, 8.294e-11, 9.26e-11, 1.039e-10,  
02317 1.156e-10, 1.297e-10, 1.46e-10, 1.641e-10, 1.858e-10, 2.1e-10,  
02318 2.383e-10, 2.724e-10, 3.116e-10, 3.538e-10, 4.173e-10, 4.727e-10,  
02319 5.503e-10, 6.337e-10, 7.32e-10, 8.298e-10, 9.328e-10, 1.059e-9,  
02320 1.176e-9, 1.328e-9, 1.445e-9, 1.593e-9, 1.77e-9, 1.954e-9,  
02321 2.175e-9, 2.405e-9, 2.622e-9, 2.906e-9, 3.294e-9, 3.713e-9,  
02322 3.98e-9, 4.384e-9, 4.987e-9, 5.311e-9, 5.874e-9, 6.337e-9,  
02323 7.027e-9, 7.39e-9, 7.769e-9, 8.374e-9, 8.605e-9, 9.165e-9,  
02324 9.415e-9, 9.511e-9, 9.704e-9, 9.588e-9, 9.45e-9, 9.086e-9,  
02325 8.798e-9, 8.469e-9, 7.697e-9, 7.168e-9, 6.255e-9, 5.772e-9,  
02326 4.97e-9, 4.271e-9, 3.653e-9, 3.154e-9, 2.742e-9, 2.435e-9,  
02327 2.166e-9, 1.936e-9, 1.731e-9, 1.556e-9, 1.399e-9, 1.272e-9,  
02328 1.157e-9, 1.066e-9, 9.844e-10, 9.258e-10, 8.787e-10, 8.421e-10,  
02329 8.083e-10, 8.046e-10, 8.067e-10, 8.181e-10, 8.325e-10, 8.517e-10,  
02330 9.151e-10, 9.351e-10, 9.677e-10, 1.071e-9, 1.126e-9, 1.219e-9,  
02331 1.297e-9, 1.408e-9, 1.476e-9, 1.517e-9, 1.6e-9, 1.649e-9,  
02332 1.678e-9, 1.746e-9, 1.742e-9, 1.728e-9, 1.699e-9, 1.655e-9,  
02333 1.561e-9, 1.48e-9, 1.451e-9, 1.411e-9, 1.171e-9, 1.106e-9,  
02334 9.714e-10, 8.523e-10, 7.346e-10, 6.241e-10, 5.371e-10, 4.704e-10,  
02335 4.144e-10, 3.683e-10, 3.292e-10, 2.942e-10, 2.62e-10, 2.341e-10,  
02336 2.104e-10, 1.884e-10, 1.7e-10, 1.546e-10, 1.394e-10, 1.265e-10,  
02337 1.14e-10, 1.019e-10, 9.279e-11, 8.283e-11, 7.458e-11, 6.668e-11,  
02338 5.976e-11, 5.33e-11, 4.794e-11, 4.289e-11, 3.841e-11, 3.467e-11,  
02339 3.13e-11, 2.832e-11, 2.582e-11, 2.356e-11, 2.152e-11, 1.97e-11,  
02340 1.808e-11, 1.664e-11, 1.539e-11, 1.434e-11, 1.344e-11, 1.269e-11,  
02341 1.209e-11, 1.162e-11, 1.129e-11, 1.108e-11, 1.099e-11, 1.103e-11,  
02342 1.119e-11, 1.148e-11, 1.193e-11, 1.252e-11, 1.329e-11, 1.421e-11,  
02343 1.555e-11, 1.685e-11, 1.839e-11, 2.054e-11, 2.317e-11, 2.571e-11,  
02344 2.839e-11, 3.171e-11, 3.49e-11, 3.886e-11, 4.287e-11, 4.645e-11,  
02345 5.047e-11, 5.592e-11, 6.109e-11, 6.628e-11, 7.381e-11, 8.088e-11,  
02346 8.966e-11, 1.045e-10, 1.12e-10, 1.287e-10, 1.486e-10, 1.662e-10,  
02347 1.866e-10, 2.133e-10, 2.524e-10, 2.776e-10, 3.204e-10, 3.559e-10,  
02348 4.028e-10, 4.448e-10, 4.882e-10, 5.244e-10, 5.605e-10, 6.018e-10,  
02349 6.328e-10, 6.579e-10, 6.541e-10, 7.024e-10, 7.074e-10, 7.068e-10,  
02350 7.009e-10, 6.698e-10, 6.545e-10, 6.209e-10, 5.834e-10, 5.412e-10,  
02351 5.001e-10, 4.231e-10, 3.727e-10, 3.211e-10, 2.833e-10, 2.447e-10,  
02352 2.097e-10, 1.843e-10, 1.639e-10, 1.449e-10, 1.27e-10, 1.161e-10,  
02353 1.033e-10, 9.282e-11, 8.407e-11, 7.639e-11, 7.023e-11, 6.474e-11,  
02354 6.142e-11, 5.76e-11, 5.568e-11, 5.472e-11, 5.39e-11, 5.455e-11,  
02355 5.54e-11, 5.587e-11, 6.23e-11, 6.49e-11, 6.868e-11, 7.382e-11,  
02356 8.022e-11, 8.372e-11, 9.243e-11, 1.004e-10, 1.062e-10, 1.13e-10,  
02357 1.176e-10, 1.244e-10, 1.279e-10, 1.298e-10, 1.302e-10, 1.312e-10,  
02358 1.295e-10, 1.244e-10, 1.211e-10, 1.167e-10, 1.098e-10, 9.927e-11,  
02359 8.854e-11, 8.011e-11, 7.182e-11, 5.923e-11, 5.212e-11, 4.453e-11,  
02360 3.832e-11, 3.371e-11, 2.987e-11, 2.651e-11, 2.354e-11, 2.093e-11,  
02361 1.863e-11, 1.662e-11, 1.486e-11, 1.331e-11, 1.193e-11, 1.071e-11,  
02362 9.628e-12, 8.66e-12, 7.801e-12, 7.031e-12, 6.347e-12, 5.733e-12,  
02363 5.182e-12, 4.695e-12, 4.26e-12, 3.874e-12, 3.533e-12, 3.235e-12,  
02364 2.979e-12, 2.76e-12, 2.579e-12, 2.432e-12, 2.321e-12, 2.246e-12,  
02365 2.205e-12, 2.196e-12, 2.223e-12, 2.288e-12, 2.387e-12, 2.525e-12,  
02366 2.704e-12, 2.925e-12, 3.191e-12, 3.508e-12, 3.876e-12, 4.303e-12,  
02367 4.793e-12, 5.347e-12, 5.978e-12, 6.682e-12, 7.467e-12, 8.34e-12,  
02368 9.293e-12, 1.035e-11, 1.152e-11, 1.285e-11, 1.428e-11, 1.586e-11,  
02369 1.764e-11, 1.972e-11, 2.214e-11, 2.478e-11, 2.776e-11, 3.151e-11,  
02370 3.591e-11, 4.103e-11, 4.66e-11, 5.395e-11, 6.306e-11, 7.172e-11,  
02371 8.358e-11, 9.67e-11, 1.11e-10, 1.325e-10, 1.494e-10, 1.736e-10,  
02372 2.007e-10, 2.296e-10, 2.608e-10, 3.004e-10, 3.361e-10, 3.727e-10,  
02373 4.373e-10, 4.838e-10, 5.483e-10, 6.006e-10, 6.535e-10, 6.899e-10,  
02374 7.687e-10, 8.444e-10, 8.798e-10, 9.135e-10, 9.532e-10, 9.757e-10,  
02375 9.968e-10, 1.006e-9, 9.949e-10, 9.789e-10, 9.564e-10, 9.215e-10,  
02376 8.51e-10, 8.394e-10, 7.707e-10, 7.152e-10, 6.274e-10, 5.598e-10,  
02377 5.028e-10, 4.3e-10, 3.71e-10, 3.245e-10, 2.809e-10, 2.461e-10,  
02378 2.154e-10, 1.91e-10, 1.685e-10, 1.487e-10, 1.313e-10, 1.163e-10,  
02379 1.031e-10, 9.172e-11, 8.221e-11, 7.382e-11, 6.693e-11, 6.079e-11,  
02380 5.581e-11, 5.167e-11, 4.811e-11, 4.506e-11, 4.255e-11, 4.083e-11,  
02381 3.949e-11, 3.881e-11, 3.861e-11, 3.858e-11, 3.951e-11, 4.045e-11,  
02382 4.24e-11, 4.487e-11, 4.806e-11, 5.133e-11, 5.518e-11, 5.919e-11,  
02383 6.533e-11, 7.031e-11, 7.762e-11, 8.305e-11, 9.252e-11, 9.727e-11,  
02384 1.045e-10, 1.117e-10, 1.2e-10, 1.275e-10, 1.341e-10, 1.362e-10,  
02385 1.438e-10, 1.45e-10, 1.455e-10, 1.455e-10, 1.434e-10, 1.381e-10,  
02386 1.301e-10, 1.276e-10, 1.163e-10, 1.089e-10, 9.911e-11, 8.943e-11,  
02387 7.618e-11, 6.424e-11, 5.717e-11, 4.866e-11, 4.257e-11, 3.773e-11,  
02388 3.331e-11, 2.958e-11, 2.629e-11, 2.316e-11, 2.073e-11, 1.841e-11,  
02389 1.635e-11, 1.464e-11, 1.31e-11, 1.16e-11, 1.047e-11, 9.408e-12,  
02390 8.414e-12, 7.521e-12, 6.705e-12, 5.993e-12, 5.371e-12, 4.815e-12,  
02391 4.338e-12, 3.921e-12, 3.567e-12, 3.265e-12, 3.01e-12, 2.795e-12,  
02392 2.613e-12, 2.464e-12, 2.346e-12, 2.256e-12, 2.195e-12, 2.165e-12,  
02393 2.166e-12, 2.198e-12, 2.262e-12, 2.364e-12, 2.502e-12, 2.682e-12,  
02394 2.908e-12, 3.187e-12, 3.533e-12, 3.946e-12, 4.418e-12, 5.013e-12,  
02395 5.708e-12, 6.379e-12, 7.43e-12, 8.39e-12, 9.51e-12, 1.078e-11,  
02396 1.259e-11, 1.438e-11, 1.63e-11, 1.814e-11, 2.055e-11, 2.348e-11,  
02397 2.664e-11, 2.956e-11, 3.3e-11, 3.677e-11, 4.032e-11, 4.494e-11,

```
02398 4.951e-11, 5.452e-11, 6.014e-11, 6.5e-11, 6.915e-11, 7.45e-11,
02399 7.971e-11, 8.468e-11, 8.726e-11, 8.995e-11, 9.182e-11, 9.509e-11,
02400 9.333e-11, 9.386e-11, 9.457e-11, 9.21e-11, 9.019e-11, 8.68e-11,
02401 8.298e-11, 7.947e-11, 7.46e-11, 7.082e-11, 6.132e-11, 5.855e-11,
02402 5.073e-11, 4.464e-11, 3.825e-11, 3.375e-11, 2.911e-11, 2.535e-11,
02403 2.16e-11, 1.907e-11, 1.665e-11, 1.463e-11, 1.291e-11, 1.133e-11,
02404 9.997e-12, 8.836e-12, 7.839e-12, 6.943e-12, 6.254e-12, 5.6e-12,
02405 5.029e-12, 4.529e-12, 4.102e-12, 3.737e-12, 3.428e-12, 3.169e-12,
02406 2.959e-12, 2.798e-12, 2.675e-12, 2.582e-12, 2.644e-12, 2.557e-12,
02407 2.614e-12, 2.717e-12, 2.874e-12, 3.056e-12, 3.187e-12, 3.631e-12,
02408 3.979e-12, 4.248e-12, 4.817e-12, 5.266e-12, 5.836e-12, 6.365e-12,
02409 6.807e-12, 7.47e-12, 7.951e-12, 8.636e-12, 8.972e-12, 9.314e-12,
02410 9.445e-12, 1.003e-11, 1.013e-11, 9.937e-12, 9.729e-12, 9.064e-12,
02411 9.119e-12, 9.124e-12, 8.704e-12, 8.078e-12, 7.47e-12, 6.329e-12,
02412 5.674e-12, 4.808e-12, 4.119e-12, 3.554e-12, 3.103e-12, 2.731e-12,
02413 2.415e-12, 2.15e-12, 1.926e-12, 1.737e-12, 1.578e-12, 1.447e-12,
02414 1.34e-12, 1.255e-12, 1.191e-12, 1.146e-12, 1.121e-12, 1.114e-12,
02415 1.126e-12, 1.156e-12, 1.207e-12, 1.278e-12, 1.372e-12, 1.49e-12,
02416 1.633e-12, 1.805e-12, 2.01e-12, 2.249e-12, 2.528e-12, 2.852e-12,
02417 3.228e-12, 3.658e-12, 4.153e-12, 4.728e-12, 5.394e-12, 6.176e-12,
02418 7.126e-12, 8.188e-12, 9.328e-12, 1.103e-11, 1.276e-11, 1.417e-11,
02419 1.615e-11, 1.84e-11, 2.155e-11, 2.429e-11, 2.826e-11, 3.222e-11,
02420 3.664e-11, 4.14e-11, 4.906e-11, 5.536e-11, 6.327e-11, 7.088e-11,
02421 8.316e-11, 9.242e-11, 1.07e-10, 1.223e-10, 1.341e-10, 1.553e-10,
02422 1.703e-10, 1.9e-10, 2.022e-10, 2.233e-10, 2.345e-10, 2.438e-10,
02423 2.546e-10, 2.599e-10, 2.661e-10, 2.703e-10, 2.686e-10, 2.662e-10,
02424 2.56e-10, 2.552e-10, 2.378e-10, 2.252e-10, 2.146e-10, 1.885e-10,
02425 1.668e-10, 1.441e-10, 1.295e-10, 1.119e-10, 9.893e-11, 8.687e-11,
02426 7.678e-11, 6.685e-11, 5.879e-11, 5.127e-11, 4.505e-11, 3.997e-11,
02427 3.511e-11
02428 };
02429
02430 static double h2ofrn[2001] = { .01095, .01126, .01205, .01322, .0143,
02431 .01506, .01548, .01534, .01486, .01373, .01262, .01134, .01001,
02432 .008702, .007475, .006481, .00548, .0046, .003833, .00311,
02433 .002543, .002049, .00168, .001374, .001046, 8.193e-4, 6.267e-4,
02434 4.968e-4, 3.924e-4, 2.983e-4, 2.477e-4, 1.997e-4, 1.596e-4,
02435 1.331e-4, 1.061e-4, 8.942e-5, 7.168e-5, 5.887e-5, 4.848e-5,
02436 3.817e-5, 3.17e-5, 2.579e-5, 2.162e-5, 1.768e-5, 1.49e-5,
02437 1.231e-5, 1.013e-5, 8.555e-6, 7.328e-6, 6.148e-6, 5.207e-6,
02438 4.387e-6, 3.741e-6, 3.22e-6, 2.753e-6, 2.346e-6, 1.985e-6,
02439 1.716e-6, 1.475e-6, 1.286e-6, 1.122e-6, 9.661e-7, 8.284e-7,
02440 7.057e-7, 6.119e-7, 5.29e-7, 4.571e-7, 3.948e-7, 3.432e-7,
02441 2.983e-7, 2.589e-7, 2.265e-7, 1.976e-7, 1.704e-7, 1.456e-7,
02442 1.26e-7, 1.101e-7, 9.648e-8, 8.415e-8, 7.34e-8, 6.441e-8,
02443 5.643e-8, 4.94e-8, 4.276e-8, 3.703e-8, 3.227e-8, 2.825e-8,
02444 2.478e-8, 2.174e-8, 1.898e-8, 1.664e-8, 1.458e-8, 1.278e-8,
02445 1.126e-8, 9.891e-9, 8.709e-9, 7.652e-9, 6.759e-9, 5.975e-9,
02446 5.31e-9, 4.728e-9, 4.214e-9, 3.792e-9, 3.463e-9, 3.226e-9,
02447 2.992e-9, 2.813e-9, 2.749e-9, 2.809e-9, 2.913e-9, 3.037e-9,
02448 3.413e-9, 3.738e-9, 4.189e-9, 4.808e-9, 5.978e-9, 7.088e-9,
02449 8.071e-9, 9.61e-9, 1.21e-8, 1.5e-8, 1.764e-8, 2.221e-8, 2.898e-8,
02450 3.948e-8, 5.068e-8, 6.227e-8, 7.898e-8, 1.033e-7, 1.437e-7,
02451 1.889e-7, 2.59e-7, 3.59e-7, 4.971e-7, 7.156e-7, 9.983e-7,
02452 1.381e-6, 1.929e-6, 2.591e-6, 3.453e-6, 4.57e-6, 5.93e-6,
02453 7.552e-6, 9.556e-6, 1.183e-5, 1.425e-5, 1.681e-5, 1.978e-5,
02454 2.335e-5, 2.668e-5, 3.022e-5, 3.371e-5, 3.715e-5, 3.967e-5,
02455 4.06e-5, 4.01e-5, 3.809e-5, 3.491e-5, 3.155e-5, 2.848e-5,
02456 2.678e-5, 2.66e-5, 2.811e-5, 3.071e-5, 3.294e-5, 3.459e-5,
02457 3.569e-5, 3.56e-5, 3.434e-5, 3.186e-5, 2.916e-5, 2.622e-5,
02458 2.275e-5, 1.918e-5, 1.62e-5, 1.373e-5, 1.182e-5, 1.006e-5,
02459 8.556e-6, 7.26e-6, 6.107e-6, 5.034e-6, 4.211e-6, 3.426e-6,
02460 2.865e-6, 2.446e-6, 1.998e-6, 1.628e-6, 1.242e-6, 1.005e-6,
02461 7.853e-7, 6.21e-7, 5.071e-7, 4.156e-7, 3.548e-7, 2.825e-7,
02462 2.261e-7, 1.916e-7, 1.51e-7, 1.279e-7, 1.059e-7, 9.14e-8,
02463 7.707e-8, 6.17e-8, 5.311e-8, 4.263e-8, 3.518e-8, 2.961e-8,
02464 2.457e-8, 2.119e-8, 1.712e-8, 1.439e-8, 1.201e-8, 1.003e-8,
02465 8.564e-9, 7.199e-9, 6.184e-9, 5.206e-9, 4.376e-9, 3.708e-9,
02466 3.157e-9, 2.725e-9, 2.361e-9, 2.074e-9, 1.797e-9, 1.562e-9,
02467 1.364e-9, 1.196e-9, 1.042e-9, 8.862e-10, 7.648e-10, 6.544e-10,
02468 5.609e-10, 4.791e-10, 4.108e-10, 3.531e-10, 3.038e-10, 2.618e-10,
02469 2.268e-10, 1.969e-10, 1.715e-10, 1.496e-10, 1.308e-10, 1.147e-10,
02470 1.008e-10, 8.894e-11, 7.885e-11, 7.031e-11, 6.355e-11, 5.854e-11,
02471 5.534e-11, 5.466e-11, 5.725e-11, 6.447e-11, 7.943e-11, 1.038e-10,
02472 1.437e-10, 2.04e-10, 2.901e-10, 4.051e-10, 5.556e-10, 7.314e-10,
02473 9.291e-10, 1.134e-9, 1.321e-9, 1.482e-9, 1.596e-9, 1.669e-9,
02474 1.715e-9, 1.762e-9, 1.817e-9, 1.828e-9, 1.848e-9, 1.873e-9,
02475 1.902e-9, 1.894e-9, 1.864e-9, 1.841e-9, 1.797e-9, 1.704e-9,
02476 1.559e-9, 1.382e-9, 1.187e-9, 1.001e-9, 8.468e-10, 7.265e-10,
02477 6.521e-10, 6.381e-10, 6.66e-10, 7.637e-10, 9.705e-10, 1.368e-9,
02478 1.856e-9, 2.656e-9, 3.954e-9, 5.96e-9, 8.72e-9, 1.247e-8,
02479 1.781e-8, 2.491e-8, 3.311e-8, 4.272e-8, 5.205e-8, 6.268e-8,
02480 7.337e-8, 8.277e-8, 9.185e-8, 1.004e-7, 1.091e-7, 1.159e-7,
02481 1.188e-7, 1.175e-7, 1.124e-7, 1.033e-7, 9.381e-8, 8.501e-8,
02482 7.956e-8, 7.894e-8, 8.331e-8, 9.102e-8, 9.836e-8, 1.035e-7,
02483 1.064e-7, 1.06e-7, 1.032e-7, 9.808e-8, 9.139e-8, 8.442e-8,
02484 7.641e-8, 6.881e-8, 6.161e-8, 5.404e-8, 4.804e-8, 4.446e-8,
```

02485 4.328e-8, 4.259e-8, 4.421e-8, 4.673e-8, 4.985e-8, 5.335e-8,  
02486 5.796e-8, 6.542e-8, 7.714e-8, 8.827e-8, 1.04e-7, 1.238e-7,  
02487 1.499e-7, 1.829e-7, 2.222e-7, 2.689e-7, 3.303e-7, 3.981e-7,  
02488 4.84e-7, 5.91e-7, 7.363e-7, 9.087e-7, 1.139e-6, 1.455e-6,  
02489 1.866e-6, 2.44e-6, 3.115e-6, 3.941e-6, 4.891e-6, 5.992e-6,  
02490 7.111e-6, 8.296e-6, 9.21e-6, 9.987e-6, 1.044e-5, 1.073e-5,  
02491 1.092e-5, 1.106e-5, 1.138e-5, 1.171e-5, 1.186e-5, 1.186e-5,  
02492 1.179e-5, 1.166e-5, 1.151e-5, 1.16e-5, 1.197e-5, 1.241e-5,  
02493 1.268e-5, 1.26e-5, 1.184e-5, 1.063e-5, 9.204e-6, 7.584e-6,  
02494 6.053e-6, 4.482e-6, 3.252e-6, 2.337e-6, 1.662e-6, 1.18e-6,  
02495 8.15e-7, 5.95e-7, 4.354e-7, 3.302e-7, 2.494e-7, 1.93e-7,  
02496 1.545e-7, 1.25e-7, 1.039e-7, 8.602e-8, 7.127e-8, 5.897e-8,  
02497 4.838e-8, 4.018e-8, 3.28e-8, 2.72e-8, 2.307e-8, 1.972e-8,  
02498 1.654e-8, 1.421e-8, 1.174e-8, 1.004e-8, 8.739e-9, 7.358e-9,  
02499 6.242e-9, 5.303e-9, 4.567e-9, 3.94e-9, 3.375e-9, 2.864e-9,  
02500 2.422e-9, 2.057e-9, 1.75e-9, 1.505e-9, 1.294e-9, 1.101e-9,  
02501 9.401e-10, 8.018e-10, 6.903e-10, 5.965e-10, 5.087e-10, 4.364e-10,  
02502 3.759e-10, 3.247e-10, 2.809e-10, 2.438e-10, 2.123e-10, 1.853e-10,  
02503 1.622e-10, 1.426e-10, 1.26e-10, 1.125e-10, 1.022e-10, 9.582e-11,  
02504 9.388e-11, 9.801e-11, 1.08e-10, 1.276e-10, 1.551e-10, 1.903e-10,  
02505 2.291e-10, 2.724e-10, 3.117e-10, 3.4e-10, 3.562e-10, 3.625e-10,  
02506 3.619e-10, 3.429e-10, 3.221e-10, 2.943e-10, 2.645e-10, 2.338e-10,  
02507 2.062e-10, 1.901e-10, 1.814e-10, 1.827e-10, 1.906e-10, 1.984e-10,  
02508 2.04e-10, 2.068e-10, 2.075e-10, 2.018e-10, 1.959e-10, 1.897e-10,  
02509 1.852e-10, 1.791e-10, 1.696e-10, 1.634e-10, 1.598e-10, 1.561e-10,  
02510 1.518e-10, 1.443e-10, 1.377e-10, 1.346e-10, 1.342e-10, 1.375e-10,  
02511 1.525e-10, 1.767e-10, 2.108e-10, 2.524e-10, 2.981e-10, 3.477e-10,  
02512 4.262e-10, 5.326e-10, 6.646e-10, 8.321e-10, 1.069e-9, 1.386e-9,  
02513 1.743e-9, 2.216e-9, 2.808e-9, 3.585e-9, 4.552e-9, 5.907e-9,  
02514 7.611e-9, 9.774e-9, 1.255e-8, 1.666e-8, 2.279e-8, 3.221e-8,  
02515 4.531e-8, 6.4e-8, 9.187e-8, 1.295e-7, 1.825e-7, 2.431e-7,  
02516 3.181e-7, 4.009e-7, 4.941e-7, 5.88e-7, 6.623e-7, 7.155e-7,  
02517 7.451e-7, 7.594e-7, 7.541e-7, 7.467e-7, 7.527e-7, 7.935e-7,  
02518 8.461e-7, 8.954e-7, 9.364e-7, 9.843e-7, 1.024e-6, 1.05e-6,  
02519 1.059e-6, 1.074e-6, 1.072e-6, 1.043e-6, 9.789e-7, 8.803e-7,  
02520 7.662e-7, 6.378e-7, 5.133e-7, 3.958e-7, 2.914e-7, 2.144e-7,  
02521 1.57e-7, 1.14e-7, 8.47e-8, 6.2e-8, 4.657e-8, 3.559e-8, 2.813e-8,  
02522 2.222e-8, 1.769e-8, 1.391e-8, 1.125e-8, 9.186e-9, 7.704e-9,  
02523 6.447e-9, 5.381e-9, 4.442e-9, 3.669e-9, 3.057e-9, 2.564e-9,  
02524 2.153e-9, 1.784e-9, 1.499e-9, 1.281e-9, 1.082e-9, 9.304e-10,  
02525 8.169e-10, 6.856e-10, 5.866e-10, 5.043e-10, 4.336e-10, 3.731e-10,  
02526 3.175e-10, 2.745e-10, 2.374e-10, 2.007e-10, 1.737e-10, 1.508e-10,  
02527 1.302e-10, 1.13e-10, 9.672e-11, 8.375e-11, 7.265e-11, 6.244e-11,  
02528 5.343e-11, 4.654e-11, 3.975e-11, 3.488e-11, 3.097e-11, 2.834e-11,  
02529 2.649e-11, 2.519e-11, 2.462e-11, 2.443e-11, 2.44e-11, 2.398e-11,  
02530 2.306e-11, 2.183e-11, 2.021e-11, 1.821e-11, 1.599e-11, 1.403e-11,  
02531 1.196e-11, 1.023e-11, 8.728e-12, 7.606e-12, 6.941e-12, 6.545e-12,  
02532 6.484e-12, 6.6e-12, 6.718e-12, 6.785e-12, 6.746e-12, 6.724e-12,  
02533 6.764e-12, 6.995e-12, 7.144e-12, 7.32e-12, 7.33e-12, 7.208e-12,  
02534 6.789e-12, 6.09e-12, 5.337e-12, 4.62e-12, 4.037e-12, 3.574e-12,  
02535 3.311e-12, 3.346e-12, 3.566e-12, 3.836e-12, 4.076e-12, 4.351e-12,  
02536 4.691e-12, 5.114e-12, 5.427e-12, 6.167e-12, 7.436e-12, 8.842e-12,  
02537 1.038e-11, 1.249e-11, 1.54e-11, 1.915e-11, 2.48e-11, 3.256e-11,  
02538 4.339e-11, 5.611e-11, 7.519e-11, 1.037e-10, 1.409e-10, 1.883e-10,  
02539 2.503e-10, 3.38e-10, 4.468e-10, 5.801e-10, 7.335e-10, 8.98e-10,  
02540 1.11e-9, 1.363e-9, 1.677e-9, 2.104e-9, 2.681e-9, 3.531e-9,  
02541 4.621e-9, 6.106e-9, 8.154e-9, 1.046e-8, 1.312e-8, 1.607e-8,  
02542 1.948e-8, 2.266e-8, 2.495e-8, 2.655e-8, 2.739e-8, 2.739e-8,  
02543 2.662e-8, 2.589e-8, 2.59e-8, 2.664e-8, 2.833e-8, 3.023e-8,  
02544 3.305e-8, 3.558e-8, 3.793e-8, 3.961e-8, 4.056e-8, 4.102e-8,  
02545 4.025e-8, 3.917e-8, 3.706e-8, 3.493e-8, 3.249e-8, 3.096e-8,  
02546 3.011e-8, 3.111e-8, 3.395e-8, 3.958e-8, 4.875e-8, 6.066e-8,  
02547 7.915e-8, 1.011e-7, 1.3e-7, 1.622e-7, 2.003e-7, 2.448e-7,  
02548 2.863e-7, 3.317e-7, 3.655e-7, 3.96e-7, 4.098e-7, 4.168e-7,  
02549 4.198e-7, 4.207e-7, 4.289e-7, 4.384e-7, 4.471e-7, 4.524e-7,  
02550 4.574e-7, 4.633e-7, 4.785e-7, 5.028e-7, 5.371e-7, 5.727e-7,  
02551 5.955e-7, 5.998e-7, 5.669e-7, 5.082e-7, 4.397e-7, 3.596e-7,  
02552 2.814e-7, 2.074e-7, 1.486e-7, 1.057e-7, 7.25e-8, 4.946e-8,  
02553 3.43e-8, 2.447e-8, 1.793e-8, 1.375e-8, 1.096e-8, 9.091e-9,  
02554 7.709e-9, 6.631e-9, 5.714e-9, 4.886e-9, 4.205e-9, 3.575e-9,  
02555 3.07e-9, 2.631e-9, 2.284e-9, 2.002e-9, 1.745e-9, 1.509e-9,  
02556 1.284e-9, 1.084e-9, 9.163e-10, 7.663e-10, 6.346e-10, 5.283e-10,  
02557 4.354e-10, 3.59e-10, 2.982e-10, 2.455e-10, 2.033e-10, 1.696e-10,  
02558 1.432e-10, 1.211e-10, 1.02e-10, 8.702e-11, 7.38e-11, 6.293e-11,  
02559 5.343e-11, 4.532e-11, 3.907e-11, 3.365e-11, 2.945e-11, 2.558e-11,  
02560 2.192e-11, 1.895e-11, 1.636e-11, 1.42e-11, 1.228e-11, 1.063e-11,  
02561 9.348e-12, 8.2e-12, 7.231e-12, 6.43e-12, 5.702e-12, 5.052e-12,  
02562 4.469e-12, 4e-12, 3.679e-12, 3.387e-12, 3.197e-12, 3.158e-12,  
02563 3.327e-12, 3.675e-12, 4.292e-12, 5.437e-12, 7.197e-12, 1.008e-11,  
02564 1.437e-11, 2.035e-11, 2.905e-11, 4.062e-11, 5.528e-11, 7.177e-11,  
02565 9.064e-11, 1.109e-10, 1.297e-10, 1.473e-10, 1.652e-10, 1.851e-10,  
02566 2.079e-10, 2.313e-10, 2.619e-10, 2.958e-10, 3.352e-10, 3.796e-10,  
02567 4.295e-10, 4.923e-10, 5.49e-10, 5.998e-10, 6.388e-10, 6.645e-10,  
02568 6.712e-10, 6.549e-10, 6.38e-10, 6.255e-10, 6.253e-10, 6.459e-10,  
02569 6.977e-10, 7.59e-10, 8.242e-10, 8.92e-10, 9.403e-10, 9.701e-10,  
02570 9.483e-10, 9.135e-10, 8.617e-10, 7.921e-10, 7.168e-10, 6.382e-10,  
02571 5.677e-10, 5.045e-10, 4.572e-10, 4.312e-10, 4.145e-10, 4.192e-10,

```
02572 4.541e-10, 5.368e-10, 6.771e-10, 8.962e-10, 1.21e-9, 1.659e-9,
02573 2.33e-9, 3.249e-9, 4.495e-9, 5.923e-9, 7.642e-9, 9.607e-9,
02574 1.178e-8, 1.399e-8, 1.584e-8, 1.73e-8, 1.816e-8, 1.87e-8,
02575 1.868e-8, 1.87e-8, 1.884e-8, 1.99e-8, 2.15e-8, 2.258e-8,
02576 2.364e-8, 2.473e-8, 2.602e-8, 2.689e-8, 2.731e-8, 2.816e-8,
02577 2.859e-8, 2.859e-8, 2.703e-8, 2.451e-8, 2.149e-8, 1.787e-8,
02578 1.449e-8, 1.111e-8, 8.282e-9, 6.121e-9, 4.494e-9, 3.367e-9,
02579 2.487e-9, 1.885e-9, 1.503e-9, 1.249e-9, 1.074e-9, 9.427e-10,
02580 8.439e-10, 7.563e-10, 6.772e-10, 6.002e-10, 5.254e-10, 4.588e-10,
02581 3.977e-10, 3.449e-10, 3.003e-10, 2.624e-10, 2.335e-10, 2.04e-10,
02582 1.771e-10, 1.534e-10, 1.296e-10, 1.097e-10, 9.173e-11, 7.73e-11,
02583 6.547e-11, 5.191e-11, 4.198e-11, 3.361e-11, 2.732e-11, 2.244e-11,
02584 1.791e-11, 1.509e-11, 1.243e-11, 1.035e-11, 8.969e-12, 7.394e-12,
02585 6.323e-12, 5.282e-12, 4.543e-12, 3.752e-12, 3.14e-12, 2.6e-12,
02586 2.194e-12, 1.825e-12, 1.511e-12, 1.245e-12, 1.024e-12, 8.539e-13,
02587 7.227e-13, 6.102e-13, 5.189e-13, 4.43e-13, 3.774e-13, 3.236e-13,
02588 2.8e-13, 2.444e-13, 2.156e-13, 1.932e-13, 1.775e-13, 1.695e-13,
02589 1.672e-13, 1.704e-13, 1.825e-13, 2.087e-13, 2.614e-13, 3.377e-13,
02590 4.817e-13, 6.989e-13, 1.062e-12, 1.562e-12, 2.288e-12, 3.295e-12,
02591 4.55e-12, 5.965e-12, 7.546e-12, 9.395e-12, 1.103e-11, 1.228e-11,
02592 1.318e-11, 1.38e-11, 1.421e-11, 1.39e-11, 1.358e-11, 1.336e-11,
02593 1.342e-11, 1.356e-11, 1.424e-11, 1.552e-11, 1.73e-11, 1.951e-11,
02594 2.128e-11, 2.249e-11, 2.277e-11, 2.226e-11, 2.111e-11, 1.922e-11,
02595 1.775e-11, 1.661e-11, 1.547e-11, 1.446e-11, 1.323e-11, 1.21e-11,
02596 1.054e-11, 9.283e-12, 8.671e-12, 8.67e-12, 9.429e-12, 1.062e-11,
02597 1.255e-11, 1.506e-11, 1.818e-11, 2.26e-11, 2.831e-11, 3.723e-11,
02598 5.092e-11, 6.968e-11, 9.826e-11, 1.349e-10, 1.87e-10, 2.58e-10,
02599 3.43e-10, 4.424e-10, 5.521e-10, 6.812e-10, 8.064e-10, 9.109e-10,
02600 9.839e-10, 1.028e-9, 1.044e-9, 1.029e-9, 1.005e-9, 1.002e-9,
02601 1.038e-9, 1.122e-9, 1.233e-9, 1.372e-9, 1.524e-9, 1.665e-9,
02602 1.804e-9, 1.908e-9, 2.015e-9, 2.117e-9, 2.219e-9, 2.336e-9,
02603 2.531e-9, 2.805e-9, 3.189e-9, 3.617e-9, 4.208e-9, 4.911e-9,
02604 5.619e-9, 6.469e-9, 7.188e-9, 7.957e-9, 8.503e-9, 9.028e-9,
02605 9.571e-9, 9.99e-9, 1.055e-8, 1.102e-8, 1.132e-8, 1.141e-8,
02606 1.145e-8, 1.145e-8, 1.176e-8, 1.224e-8, 1.304e-8, 1.388e-8,
02607 1.445e-8, 1.453e-8, 1.368e-8, 1.22e-8, 1.042e-8, 8.404e-9,
02608 6.403e-9, 4.643e-9, 3.325e-9, 2.335e-9, 1.638e-9, 1.19e-9,
02609 9.161e-10, 7.412e-10, 6.226e-10, 5.516e-10, 5.068e-10, 4.831e-10,
02610 4.856e-10, 5.162e-10, 5.785e-10, 6.539e-10, 7.485e-10, 8.565e-10,
02611 9.534e-10, 1.052e-9, 1.115e-9, 1.173e-9, 1.203e-9, 1.224e-9,
02612 1.243e-9, 1.248e-9, 1.261e-9, 1.265e-9, 1.25e-9, 1.217e-9,
02613 1.176e-9, 1.145e-9, 1.153e-9, 1.199e-9, 1.278e-9, 1.366e-9,
02614 1.426e-9, 1.444e-9, 1.365e-9, 1.224e-9, 1.051e-9, 8.539e-10,
02615 6.564e-10, 4.751e-10, 3.404e-10, 2.377e-10, 1.631e-10, 1.114e-10,
02616 7.87e-11, 5.793e-11, 4.284e-11, 3.3e-11, 2.62e-11, 2.152e-11,
02617 1.777e-11, 1.496e-11, 1.242e-11, 1.037e-11, 8.725e-12, 7.004e-12,
02618 5.718e-12, 4.769e-12, 3.952e-12, 3.336e-12, 2.712e-12, 2.213e-12,
02619 1.803e-12, 1.492e-12, 1.236e-12, 1.006e-12, 8.384e-13, 7.063e-13,
02620 5.879e-13, 4.93e-13, 4.171e-13, 3.569e-13, 3.083e-13, 2.688e-13,
02621 2.333e-13, 2.035e-13, 1.82e-13, 1.682e-13, 1.635e-13, 1.674e-13,
02622 1.769e-13, 2.022e-13, 2.485e-13, 3.127e-13, 4.25e-13, 5.928e-13,
02623 8.514e-13, 1.236e-12, 1.701e-12, 2.392e-12, 3.231e-12, 4.35e-12,
02624 5.559e-12, 6.915e-12, 8.519e-12, 1.013e-11, 1.146e-11, 1.24e-11,
02625 1.305e-11, 1.333e-11, 1.318e-11, 1.263e-11, 1.238e-11, 1.244e-11,
02626 1.305e-11, 1.432e-11, 1.623e-11, 1.846e-11, 2.09e-11, 2.328e-11,
02627 2.526e-11, 2.637e-11, 2.702e-11, 2.794e-11, 2.889e-11, 2.989e-11,
02628 3.231e-11, 3.68e-11, 4.375e-11, 5.504e-11, 7.159e-11, 9.502e-11,
02629 1.279e-10, 1.645e-10, 2.098e-10, 2.618e-10, 3.189e-10, 3.79e-10,
02630 4.303e-10, 4.753e-10, 5.027e-10, 5.221e-10, 5.293e-10, 5.346e-10,
02631 5.467e-10, 5.796e-10, 6.2e-10, 6.454e-10, 6.705e-10, 6.925e-10,
02632 7.233e-10, 7.35e-10, 7.538e-10, 7.861e-10, 8.077e-10, 8.132e-10,
02633 7.749e-10, 7.036e-10, 6.143e-10, 5.093e-10, 4.089e-10, 3.092e-10,
02634 2.299e-10, 1.705e-10, 1.277e-10, 9.723e-11, 7.533e-11, 6.126e-11,
02635 5.154e-11, 4.428e-11, 3.913e-11, 3.521e-11, 3.297e-11, 3.275e-11,
02636 3.46e-11, 3.798e-11, 4.251e-11, 4.745e-11, 5.232e-11, 5.606e-11,
02637 5.82e-11, 5.88e-11, 5.79e-11, 5.661e-11, 5.491e-11, 5.366e-11,
02638 5.341e-11, 5.353e-11, 5.336e-11, 5.293e-11, 5.248e-11, 5.235e-11,
02639 5.208e-11, 5.322e-11, 5.521e-11, 5.725e-11, 5.827e-11, 5.685e-11,
02640 5.245e-11, 4.612e-11, 3.884e-11, 3.129e-11, 2.404e-11, 1.732e-11,
02641 1.223e-11, 8.574e-12, 5.888e-12, 3.986e-12, 2.732e-12, 1.948e-12,
02642 1.414e-12, 1.061e-12, 8.298e-13, 6.612e-13, 5.413e-13, 4.472e-13,
02643 3.772e-13, 3.181e-13, 2.645e-13, 2.171e-13, 1.778e-13, 1.464e-13,
02644 1.183e-13, 9.637e-14, 7.991e-14, 6.668e-14, 5.57e-14, 4.663e-14,
02645 3.848e-14, 3.233e-14, 2.706e-14, 2.284e-14, 1.944e-14, 1.664e-14,
02646 1.43e-14, 1.233e-14, 1.066e-14, 9.234e-15, 8.023e-15, 6.993e-15,
02647 6.119e-15, 5.384e-15, 4.774e-15, 4.283e-15, 3.916e-15, 3.695e-15,
02648 3.682e-15, 4.004e-15, 4.912e-15, 6.853e-15, 1.056e-14, 1.712e-14,
02649 2.804e-14, 4.516e-14, 7.113e-14, 1.084e-13, 1.426e-13, 1.734e-13,
02650 1.978e-13, 2.194e-13, 2.388e-13, 2.489e-13, 2.626e-13, 2.865e-13,
02651 3.105e-13, 3.387e-13, 3.652e-13, 3.984e-13, 4.398e-13, 4.906e-13,
02652 5.55e-13, 6.517e-13, 7.813e-13, 9.272e-13, 1.164e-12, 1.434e-12,
02653 1.849e-12, 2.524e-12, 3.328e-12, 4.523e-12, 6.108e-12, 8.207e-12,
02654 1.122e-11, 1.477e-11, 1.9e-11, 2.412e-11, 2.984e-11, 3.68e-11,
02655 4.353e-11, 4.963e-11, 5.478e-11, 5.903e-11, 6.233e-11, 6.483e-11,
02656 6.904e-11, 7.569e-11, 8.719e-11, 1.048e-10, 1.278e-10, 1.557e-10,
02657 1.869e-10, 2.218e-10, 2.61e-10, 2.975e-10, 3.371e-10, 3.746e-10,
02658 4.065e-10, 4.336e-10, 4.503e-10, 4.701e-10, 4.8e-10, 4.917e-10,
```

02659 5.038e-10, 5.128e-10, 5.143e-10, 5.071e-10, 5.019e-10, 5.025e-10,  
02660 5.183e-10, 5.496e-10, 5.877e-10, 6.235e-10, 6.42e-10, 6.234e-10,  
02661 5.698e-10, 4.916e-10, 4.022e-10, 3.126e-10, 2.282e-10, 1.639e-10,  
02662 1.142e-10, 7.919e-11, 5.69e-11, 4.313e-11, 3.413e-11, 2.807e-11,  
02663 2.41e-11, 2.166e-11, 2.024e-11, 1.946e-11, 1.929e-11, 1.963e-11,  
02664 2.035e-11, 2.162e-11, 2.305e-11, 2.493e-11, 2.748e-11, 3.048e-11,  
02665 3.413e-11, 3.754e-11, 4.155e-11, 4.635e-11, 5.11e-11, 5.734e-11,  
02666 6.338e-11, 6.99e-11, 7.611e-11, 8.125e-11, 8.654e-11, 8.951e-11,  
02667 9.182e-11, 9.31e-11, 9.273e-11, 9.094e-11, 8.849e-11, 8.662e-11,  
02668 8.67e-11, 8.972e-11, 9.566e-11, 1.025e-10, 1.083e-10, 1.111e-10,  
02669 1.074e-10, 9.771e-11, 8.468e-11, 6.958e-11, 5.47e-11, 4.04e-11,  
02670 2.94e-11, 2.075e-11, 1.442e-11, 1.01e-11, 7.281e-12, 5.409e-12,  
02671 4.138e-12, 3.304e-12, 2.784e-12, 2.473e-12, 2.273e-12, 2.186e-12,  
02672 2.118e-12, 2.066e-12, 1.958e-12, 1.818e-12, 1.675e-12, 1.509e-12,  
02673 1.349e-12, 1.171e-12, 9.838e-13, 8.213e-13, 6.765e-13, 5.378e-13,  
02674 4.161e-13, 3.119e-13, 2.279e-13, 1.637e-13, 1.152e-13, 8.112e-14,  
02675 5.919e-14, 4.47e-14, 3.492e-14, 2.811e-14, 2.319e-14, 1.948e-14,  
02676 1.66e-14, 1.432e-14, 1.251e-14, 1.109e-14, 1.006e-14, 9.45e-15,  
02677 9.384e-15, 1.012e-14, 1.216e-14, 1.636e-14, 2.305e-14, 3.488e-14,  
02678 5.572e-14, 8.479e-14, 1.265e-13, 1.905e-13, 2.73e-13, 3.809e-13,  
02679 4.955e-13, 6.303e-13, 7.861e-13, 9.427e-13, 1.097e-12, 1.212e-12,  
02680 1.328e-12, 1.415e-12, 1.463e-12, 1.495e-12, 1.571e-12, 1.731e-12,  
02681 1.981e-12, 2.387e-12, 2.93e-12, 3.642e-12, 4.584e-12, 5.822e-12,  
02682 7.278e-12, 9.193e-12, 1.135e-11, 1.382e-11, 1.662e-11, 1.958e-11,  
02683 2.286e-11, 2.559e-11, 2.805e-11, 2.988e-11, 3.106e-11, 3.182e-11,  
02684 3.2e-11, 3.258e-11, 3.362e-11, 3.558e-11, 3.688e-11, 3.8e-11,  
02685 3.929e-11, 4.062e-11, 4.186e-11, 4.293e-11, 4.48e-11, 4.643e-11,  
02686 4.704e-11, 4.571e-11, 4.206e-11, 3.715e-11, 3.131e-11, 2.541e-11,  
02687 1.978e-11, 1.508e-11, 1.146e-11, 8.7e-12, 6.603e-12, 5.162e-12,  
02688 4.157e-12, 3.408e-12, 2.829e-12, 2.405e-12, 2.071e-12, 1.826e-12,  
02689 1.648e-12, 1.542e-12, 1.489e-12, 1.485e-12, 1.493e-12, 1.545e-12,  
02690 1.637e-12, 1.814e-12, 2.061e-12, 2.312e-12, 2.651e-12, 3.03e-12,  
02691 3.46e-12, 3.901e-12, 4.306e-12, 4.721e-12, 5.008e-12, 5.281e-12,  
02692 5.541e-12, 5.791e-12, 6.115e-12, 6.442e-12, 6.68e-12, 6.791e-12,  
02693 6.831e-12, 6.839e-12, 6.946e-12, 7.128e-12, 7.537e-12, 8.036e-12,  
02694 8.392e-12, 8.526e-12, 8.11e-12, 7.325e-12, 6.329e-12, 5.183e-12,  
02695 4.081e-12, 2.985e-12, 2.141e-12, 1.492e-12, 1.015e-12, 6.684e-13,  
02696 4.414e-13, 2.987e-13, 2.038e-13, 1.391e-13, 9.86e-14, 7.24e-14,  
02697 5.493e-14, 4.288e-14, 3.427e-14, 2.787e-14, 2.296e-14, 1.909e-14,  
02698 1.598e-14, 1.344e-14, 1.135e-14, 9.616e-15, 8.169e-15, 6.957e-15,  
02699 5.938e-15, 5.08e-15, 4.353e-15, 3.738e-15, 3.217e-15, 2.773e-15,  
02700 2.397e-15, 2.077e-15, 1.805e-15, 1.575e-15, 1.382e-15, 1.221e-15,  
02701 1.09e-15, 9.855e-16, 9.068e-16, 8.537e-16, 8.27e-16, 8.29e-16,  
02702 8.634e-16, 9.359e-16, 1.055e-15, 1.233e-15, 1.486e-15, 1.839e-15,  
02703 2.326e-15, 2.998e-15, 3.934e-15, 5.256e-15, 7.164e-15, 9.984e-15,  
02704 1.427e-14, 2.099e-14, 3.196e-14, 5.121e-14, 7.908e-14, 1.131e-13,  
02705 1.602e-13, 2.239e-13, 3.075e-13, 4.134e-13, 5.749e-13, 7.886e-13,  
02706 1.071e-12, 1.464e-12, 2.032e-12, 2.8e-12, 3.732e-12, 4.996e-12,  
02707 6.483e-12, 8.143e-12, 1.006e-11, 1.238e-11, 1.484e-11, 1.744e-11,  
02708 2.02e-11, 2.274e-11, 2.562e-11, 2.848e-11, 3.191e-11, 3.617e-11,  
02709 4.081e-11, 4.577e-11, 4.937e-11, 5.204e-11, 5.401e-11, 5.462e-11,  
02710 5.507e-11, 5.51e-11, 5.605e-11, 5.686e-11, 5.739e-11, 5.766e-11,  
02711 5.74e-11, 5.754e-11, 5.761e-11, 5.777e-11, 5.712e-11, 5.51e-11,  
02712 5.088e-11, 4.438e-11, 3.728e-11, 2.994e-11, 2.305e-11, 1.715e-11,  
02713 1.256e-11, 9.208e-12, 6.745e-12, 5.014e-12, 3.785e-12, 2.9e-12,  
02714 2.239e-12, 1.757e-12, 1.414e-12, 1.142e-12, 9.482e-13, 8.01e-13,  
02715 6.961e-13, 6.253e-13, 5.735e-13, 5.433e-13, 5.352e-13, 5.493e-13,  
02716 5.706e-13, 6.068e-13, 6.531e-13, 7.109e-13, 7.767e-13, 8.59e-13,  
02717 9.792e-13, 1.142e-12, 1.371e-12, 1.65e-12, 1.957e-12, 2.302e-12,  
02718 2.705e-12, 3.145e-12, 3.608e-12, 4.071e-12, 4.602e-12, 5.133e-12,  
02719 5.572e-12, 5.987e-12, 6.248e-12, 6.533e-12, 6.757e-12, 6.935e-12,  
02720 7.224e-12, 7.422e-12, 7.538e-12, 7.547e-12, 7.495e-12, 7.543e-12,  
02721 7.725e-12, 8.139e-12, 8.627e-12, 9.146e-12, 9.443e-12, 9.318e-12,  
02722 8.649e-12, 7.512e-12, 6.261e-12, 4.915e-12, 3.647e-12, 2.597e-12,  
02723 1.785e-12, 1.242e-12, 8.66e-13, 6.207e-13, 4.61e-13, 3.444e-13,  
02724 2.634e-13, 2.1e-13, 1.725e-13, 1.455e-13, 1.237e-13, 1.085e-13,  
02725 9.513e-14, 7.978e-14, 6.603e-14, 5.288e-14, 4.084e-14, 2.952e-14,  
02726 2.157e-14, 1.593e-14, 1.199e-14, 9.267e-15, 7.365e-15, 6.004e-15,  
02727 4.995e-15, 4.218e-15, 3.601e-15, 3.101e-15, 2.692e-15, 2.36e-15,  
02728 2.094e-15, 1.891e-15, 1.755e-15, 1.699e-15, 1.755e-15, 1.987e-15,  
02729 2.506e-15, 3.506e-15, 5.289e-15, 8.311e-15, 1.325e-14, 2.129e-14,  
02730 3.237e-14, 4.595e-14, 6.441e-14, 8.433e-14, 1.074e-13, 1.383e-13,  
02731 1.762e-13, 2.281e-13, 2.831e-13, 3.523e-13, 4.38e-13, 5.304e-13,  
02732 6.29e-13, 7.142e-13, 8.032e-13, 8.934e-13, 9.888e-13, 1.109e-12,  
02733 1.261e-12, 1.462e-12, 1.74e-12, 2.099e-12, 2.535e-12, 3.008e-12,  
02734 3.462e-12, 3.856e-12, 4.098e-12, 4.239e-12, 4.234e-12, 4.132e-12,  
02735 3.986e-12, 3.866e-12, 3.829e-12, 3.742e-12, 3.705e-12, 3.694e-12,  
02736 3.765e-12, 3.849e-12, 3.929e-12, 4.056e-12, 4.092e-12, 4.047e-12,  
02737 3.792e-12, 3.407e-12, 2.953e-12, 2.429e-12, 1.931e-12, 1.46e-12,  
02738 1.099e-12, 8.199e-13, 6.077e-13, 4.449e-13, 3.359e-13, 2.524e-13,  
02739 1.881e-13, 1.391e-13, 1.02e-13, 7.544e-14, 5.555e-14, 4.22e-14,  
02740 3.321e-14, 2.686e-14, 2.212e-14, 1.78e-14, 1.369e-14, 1.094e-14,  
02741 9.13e-15, 8.101e-15, 7.828e-15, 8.393e-15, 1.012e-14, 1.259e-14,  
02742 1.538e-14, 1.961e-14, 2.619e-14, 3.679e-14, 5.049e-14, 6.917e-14,  
02743 8.88e-14, 1.115e-13, 1.373e-13, 1.619e-13, 1.878e-13, 2.111e-13,  
02744 2.33e-13, 2.503e-13, 2.613e-13, 2.743e-13, 2.826e-13, 2.976e-13,  
02745 3.162e-13, 3.36e-13, 3.491e-13, 3.541e-13, 3.595e-13, 3.608e-13,

```

02746     3.709e-13, 3.869e-13, 4.12e-13, 4.366e-13, 4.504e-13, 4.379e-13,
02747     3.955e-13, 3.385e-13, 2.741e-13, 2.089e-13, 1.427e-13, 9.294e-14,
02748     5.775e-14, 3.565e-14, 2.21e-14, 1.398e-14, 9.194e-15, 6.363e-15,
02749     4.644e-15, 3.55e-15, 2.808e-15, 2.274e-15, 1.871e-15, 1.557e-15,
02750     1.308e-15, 1.108e-15, 9.488e-16, 8.222e-16, 7.238e-16, 6.506e-16,
02751     6.008e-16, 5.742e-16, 5.724e-16, 5.991e-16, 6.625e-16, 7.775e-16,
02752     9.734e-16, 1.306e-15, 1.88e-15, 2.879e-15, 4.616e-15, 7.579e-15,
02753     1.248e-14, 2.03e-14, 3.244e-14, 5.171e-14, 7.394e-14, 9.676e-14,
02754     1.199e-13, 1.467e-13, 1.737e-13, 2.02e-13, 2.425e-13, 3.016e-13,
02755     3.7e-13, 4.617e-13, 5.949e-13, 7.473e-13, 9.378e-13, 1.191e-12,
02756     1.481e-12, 1.813e-12, 2.232e-12, 2.722e-12, 3.254e-12, 3.845e-12,
02757     4.458e-12, 5.048e-12, 5.511e-12, 5.898e-12, 6.204e-12, 6.293e-12,
02758     6.386e-12, 6.467e-12, 6.507e-12, 6.466e-12, 6.443e-12, 6.598e-12,
02759     6.873e-12, 7.3e-12, 7.816e-12, 8.368e-12, 8.643e-12, 8.466e-12,
02760     7.871e-12, 6.853e-12, 5.714e-12, 4.482e-12, 3.392e-12, 2.613e-12,
02761     2.008e-12, 1.562e-12, 1.228e-12, 9.888e-13, 7.646e-13, 5.769e-13,
02762     4.368e-13, 3.324e-13, 2.508e-13, 1.916e-13
02763 };
02764
02765 static double xfcrev[15] =
02766 { 1.003, 1.009, 1.015, 1.023, 1.029, 1.033, 1.037,
02767   1.039, 1.04, 1.046, 1.036, 1.027, 1.01, 1.002, 1.
02768 };
02769
02770 double sfac;
02771
02772 /* Get H2O continuum absorption... */
02773 double xw = nu / 10 + 1;
02774 if (xw >= 1 && xw < 2001) {
02775     int iw = (int) xw;
02776     double dw = xw - iw;
02777     double ew = 1 - dw;
02778     double cw296 = ew * h2o296[iw - 1] + dw * h2o296[iw];
02779     double cw260 = ew * h2o260[iw - 1] + dw * h2o260[iw];
02780     double cwfrn = ew * h2ofrn[iw - 1] + dw * h2ofrn[iw];
02781     if (nu <= 820 || nu >= 960) {
02782         sfac = 1;
02783     } else {
02784         double xx = (nu - 820) / 10;
02785         int ix = (int) xx;
02786         double dx = xx - ix;
02787         sfac = (1 - dx) * xfcrev[ix] + dx * xfcrev[ix + 1];
02788     }
02789     double ctws1f =
02790         sfac * cw296 * pow(cw260 / cw296, (296 - t) / (296 - 260));
02791     double vf2 = POW2(nu - 370);
02792     double vf6 = POW3(vf2);
02793     double fscal = 36100 / (vf2 + vf6 * 1e-8 + 36100) * -.25 + 1;
02794     double ctwfrn = cwfrn * fscal;
02795     double a1 = nu * u * tanh(.7193876 / t * nu);
02796     double a2 = 296 / t;
02797     double a3 = p / P0 * (q * ctws1f + (1 - q) * ctwfrn) * 1e-20;
02798     return a1 * a2 * a3;
02799 } else
02800     return 0;
02801 }

```

### 5.17.2.8 ctmn2() double ctmn2 (

```

    double nu,
    double p,
    double t )

```

Compute nitrogen continuum (absorption coefficient).

Definition at line 2805 of file [jurassic.c](#).

```

02808     {
02809
02810     static double ba[98] = { 0., 4.45e-8, 5.22e-8, 6.46e-8, 7.75e-8, 9.03e-8,
02811     1.06e-7, 1.21e-7, 1.37e-7, 1.57e-7, 1.75e-7, 2.01e-7, 2.3e-7,
02812     2.59e-7, 2.95e-7, 3.26e-7, 3.66e-7, 4.05e-7, 4.47e-7, 4.92e-7,
02813     5.34e-7, 5.84e-7, 6.24e-7, 6.67e-7, 7.14e-7, 7.26e-7, 7.54e-7,
02814     7.84e-7, 8.09e-7, 8.42e-7, 8.62e-7, 8.87e-7, 9.11e-7, 9.36e-7,
02815     9.76e-7, 1.03e-6, 1.11e-6, 1.23e-6, 1.39e-6, 1.61e-6, 1.76e-6,
02816     1.94e-6, 1.97e-6, 1.87e-6, 1.75e-6, 1.56e-6, 1.42e-6, 1.35e-6,
02817     1.32e-6, 1.29e-6, 1.29e-6, 1.29e-6, 1.3e-6, 1.32e-6, 1.33e-6,
02818     1.34e-6, 1.35e-6, 1.33e-6, 1.31e-6, 1.29e-6, 1.24e-6, 1.2e-6,
02819     1.16e-6, 1.1e-6, 1.04e-6, 9.96e-7, 9.38e-7, 8.63e-7, 7.98e-7,
02820     7.26e-7, 6.55e-7, 5.94e-7, 5.35e-7, 4.74e-7, 4.24e-7, 3.77e-7,

```



```

02821      3.33e-7, 2.96e-7, 2.63e-7, 2.34e-7, 2.08e-7, 1.85e-7, 1.67e-7,
02822      1.47e-7, 1.32e-7, 1.2e-7, 1.09e-7, 9.85e-8, 9.08e-8, 8.18e-8,
02823      7.56e-8, 6.85e-8, 6.14e-8, 5.83e-8, 5.77e-8, 5e-8, 4.32e-8, 0.
02824  };
02825
02826  static double betaa[98] = { 802., 802., 761., 722., 679., 646., 609., 562.,
02827      511., 472., 436., 406., 377., 355., 338., 319., 299., 278., 255.,
02828      233., 208., 184., 149., 107., 66., 25., -13., -49., -82., -104.,
02829      -119., -130., -139., -144., -146., -146., -147., -148., -150.,
02830      -153., -160., -169., -181., -189., -195., -200., -205., -209.,
02831      -211., -210., -210., -209., -205., -199., -190., -180., -168.,
02832      -157., -143., -126., -108., -89., -63., -32., 1., 35., 65., 95.,
02833      121., 141., 152., 161., 164., 164., 161., 155., 148., 143., 137.,
02834      133., 131., 133., 139., 150., 165., 187., 213., 248., 284., 321.,
02835      372., 449., 514., 569., 609., 642., 673., 673.
02836  };
02837
02838  static double nua[98] = { 2120., 2125., 2130., 2135., 2140., 2145., 2150.,
02839      2155., 2160., 2165., 2170., 2175., 2180., 2185., 2190., 2195.,
02840      2200., 2205., 2210., 2215., 2220., 2225., 2230., 2235., 2240.,
02841      2245., 2250., 2255., 2260., 2265., 2270., 2275., 2280., 2285.,
02842      2290., 2295., 2300., 2305., 2310., 2315., 2320., 2325., 2330.,
02843      2335., 2340., 2345., 2350., 2355., 2360., 2365., 2370., 2375.,
02844      2380., 2385., 2390., 2395., 2400., 2405., 2410., 2415., 2420.,
02845      2425., 2430., 2435., 2440., 2445., 2450., 2455., 2460., 2465.,
02846      2470., 2475., 2480., 2485., 2490., 2495., 2500., 2505., 2510.,
02847      2515., 2520., 2525., 2530., 2535., 2540., 2545., 2550., 2555.,
02848      2560., 2565., 2570., 2575., 2580., 2585., 2590., 2595., 2600., 2605.
02849  };
02850
02851  const double q_n2 = 0.79, t0 = 273.0, tr = 296.0;
02852
02853  /* Check wavenumber range... */
02854  if (nu < nua[0] || nu > nua[97])
02855      return 0;
02856
02857  /* Interpolate B and beta... */
02858  int idx = locate_reg(nua, 98, nu);
02859  double b = LIN(nua[idx], ba[idx], nua[idx + 1], ba[idx + 1], nu);
02860  double beta = LIN(nua[idx], betaa[idx], nua[idx + 1], betaa[idx + 1], nu);
02861
02862  /* Compute absorption coefficient... */
02863  return 0.1 * POW2(p / P0 * t0 / t) * exp(beta * (1 / tr - 1 / t))
02864      * q_n2 * b * (q_n2 + (1 - q_n2) * (1.294 - 0.4545 * t / tr));
02865 }

```

Here is the call graph for this function:



```

5.17.2.9 ctmo2() double ctmo2 (
    double nu,
    double p,
    double t )

```

Compute oxygen continuum (absorption coefficient).

Definition at line 2869 of file [jurassic.c](#).

```

02872  {
02873
02874  static double ba[90] = { 0., .061, .074, .084, .096, .12, .162, .208, .246,
02875      .285, .314, .38, .444, .5, .571, .673, .768, .853, .966, 1.097,

```

```

02876     1.214, 1.333, 1.466, 1.591, 1.693, 1.796, 1.922, 2.037, 2.154,
02877     2.264, 2.375, 2.508, 2.671, 2.847, 3.066, 3.417, 3.828, 4.204,
02878     4.453, 4.599, 4.528, 4.284, 3.955, 3.678, 3.477, 3.346, 3.29,
02879     3.251, 3.231, 3.226, 3.212, 3.192, 3.108, 3.033, 2.911, 2.798,
02880     2.646, 2.508, 2.322, 2.13, 1.928, 1.757, 1.588, 1.417, 1.253,
02881     1.109, .99, .888, .791, .678, .587, .524, .464, .403, .357, .32,
02882     .29, .267, .242, .215, .182, .16, .146, .128, .103, .087, .081,
02883     .071, .064, 0.
02884 };
02885
02886 static double betaa[90] = { 467., 467., 400., 315., 379., 368., 475., 521.,
02887     531., 512., 442., 444., 430., 381., 335., 324., 296., 248., 215.,
02888     193., 158., 127., 101., 71., 31., -6., -26., -47., -63., -79.,
02889     -88., -88., -87., -90., -98., -99., -109., -134., -160., -167.,
02890     -164., -158., -153., -151., -156., -166., -168., -173., -170.,
02891     -161., -145., -126., -108., -84., -59., -29., 4., 41., 73., 97.,
02892     123., 159., 198., 220., 242., 256., 281., 311., 334., 319., 313.,
02893     321., 323., 310., 315., 320., 335., 361., 378., 373., 338., 319.,
02894     346., 322., 291., 290., 350., 371., 504., 504.
02895 };
02896
02897 static double nua[90] = { 1360., 1365., 1370., 1375., 1380., 1385., 1390.,
02898     1395., 1400., 1405., 1410., 1415., 1420., 1425., 1430., 1435.,
02899     1440., 1445., 1450., 1455., 1460., 1465., 1470., 1475., 1480.,
02900     1485., 1490., 1495., 1500., 1505., 1510., 1515., 1520., 1525.,
02901     1530., 1535., 1540., 1545., 1550., 1555., 1560., 1565., 1570.,
02902     1575., 1580., 1585., 1590., 1595., 1600., 1605., 1610., 1615.,
02903     1620., 1625., 1630., 1635., 1640., 1645., 1650., 1655., 1660.,
02904     1665., 1670., 1675., 1680., 1685., 1690., 1695., 1700., 1705.,
02905     1710., 1715., 1720., 1725., 1730., 1735., 1740., 1745., 1750.,
02906     1755., 1760., 1765., 1770., 1775., 1780., 1785., 1790., 1795.,
02907     1800., 1805.
02908 };
02909
02910 const double q_o2 = 0.21, t0 = 273, tr = 296;
02911
02912 /* Check wavenumber range... */
02913 if (nu < nua[0] || nu > nua[89])
02914     return 0;
02915
02916 /* Interpolate B and beta... */
02917 int idx = locate_reg(nua, 90, nu);
02918 double b = LIN(nua[idx], ba[idx], nua[idx + 1], ba[idx + 1], nu);
02919 double beta = LIN(nua[idx], betaa[idx], nua[idx + 1], betaa[idx + 1], nu);
02920
02921 /* Compute absorption coefficient... */
02922 return 0.1 * POW2(p / P0 * t0 / t) * exp(beta * (1 / tr - 1 / t)) * q_o2 *
02923     b;
02924 }

```

Here is the call graph for this function:



**5.17.2.10 copy\_atm()** void copy\_atm (

```

    ctl_t * ctl,
    atm_t * atm_dest,
    atm_t * atm_src,
    int init )

```

Copy and initialize atmospheric data.

Definition at line 2928 of file [jurassic.c](#).

```

02932     {
02933
02934     /* Data size... */
02935     size_t s = (size_t) atm_src->np * sizeof(double);
02936
02937     /* Copy data... */
02938     atm_dest->np = atm_src->np;
02939     memcpy(atm_dest->time, atm_src->time, s);
02940     memcpy(atm_dest->z, atm_src->z, s);
02941     memcpy(atm_dest->lon, atm_src->lon, s);
02942     memcpy(atm_dest->lat, atm_src->lat, s);
02943     memcpy(atm_dest->p, atm_src->p, s);
02944     memcpy(atm_dest->t, atm_src->t, s);
02945     for (int ig = 0; ig < ctl->ng; ig++)
02946         memcpy(atm_dest->q[ig], atm_src->q[ig], s);
02947     for (int iw = 0; iw < ctl->nw; iw++)
02948         memcpy(atm_dest->k[iw], atm_src->k[iw], s);
02949     atm_dest->clz = atm_src->clz;
02950     atm_dest->cldz = atm_src->cldz;
02951     for (int icl = 0; icl < ctl->ncl; icl++)
02952         atm_dest->clk[icl] = atm_src->clk[icl];
02953     atm_dest->sfz = atm_src->sfz;
02954     atm_dest->sfp = atm_src->sfp;
02955     atm_dest->sft = atm_src->sft;
02956     for (int isf = 0; isf < ctl->nsf; isf++)
02957         atm_dest->sfeps[isf] = atm_src->sfeps[isf];
02958
02959     /* Initialize... */
02960     if (init)
02961         for (int ip = 0; ip < atm_dest->np; ip++) {
02962             atm_dest->p[ip] = 0;
02963             atm_dest->t[ip] = 0;
02964             for (int ig = 0; ig < ctl->ng; ig++)
02965                 atm_dest->q[ig][ip] = 0;
02966             for (int iw = 0; iw < ctl->nw; iw++)
02967                 atm_dest->k[iw][ip] = 0;
02968             atm_dest->clz = 0;
02969             atm_dest->cldz = 0;
02970             for (int icl = 0; icl < ctl->ncl; icl++)
02971                 atm_dest->clk[icl] = 0;
02972             atm_dest->sfz = 0;
02973             atm_dest->sfp = 0;
02974             atm_dest->sft = 0;
02975             for (int isf = 0; isf < ctl->nsf; isf++)
02976                 atm_dest->sfeps[isf] = 1;
02977         }
02978 }

```

**5.17.2.11 copy\_obs()** void copy\_obs (

```

    ctl_t * ctl,
    obs_t * obs_dest,
    obs_t * obs_src,
    int init )

```

Copy and initialize observation data.

Definition at line 2982 of file [jurassic.c](#).

```

02986     {
02987
02988     /* Data size... */
02989     size_t s = (size_t) obs_src->nr * sizeof(double);
02990
02991     /* Copy data... */
02992     obs_dest->nr = obs_src->nr;
02993     memcpy(obs_dest->time, obs_src->time, s);
02994     memcpy(obs_dest->obsz, obs_src->obsz, s);
02995     memcpy(obs_dest->obslon, obs_src->obslon, s);
02996     memcpy(obs_dest->obslat, obs_src->obslat, s);
02997     memcpy(obs_dest->vpz, obs_src->vpz, s);
02998     memcpy(obs_dest->vplon, obs_src->vplon, s);
02999     memcpy(obs_dest->vplat, obs_src->vplat, s);
03000     memcpy(obs_dest->tpz, obs_src->tpz, s);
03001     memcpy(obs_dest->tplon, obs_src->tplon, s);
03002     memcpy(obs_dest->tplat, obs_src->tplat, s);
03003     for (int id = 0; id < ctl->nd; id++)
03004         memcpy(obs_dest->rad[id], obs_src->rad[id], s);

```

```

03005     for (int id = 0; id < ctl->nd; id++)
03006         memcpy(obs_dest->tau[id], obs_src->tau[id], s);
03007
03008     /* Initialize... */
03009     if (init)
03010         for (int id = 0; id < ctl->nd; id++)
03011             for (int ir = 0; ir < obs_dest->nr; ir++)
03012                 if (gsl_finite(obs_dest->rad[id][ir])) {
03013                     obs_dest->rad[id][ir] = 0;
03014                     obs_dest->tau[id][ir] = 0;
03015                 }
03016 }

```

**5.17.2.12 find\_emitter()** int find\_emitter (  
     ctl\_t \* ctl,  
     const char \* emitter )

Find index of an emitter.

Definition at line 3020 of file [jurassic.c](#).

```

03022     {
03023
03024     for (int ig = 0; ig < ctl->ng; ig++)
03025         if (strcasecmp(ctl->emitter[ig], emitter) == 0)
03026             return ig;
03027
03028     return -1;
03029 }

```

**5.17.2.13 formod()** void formod (  
     ctl\_t \* ctl,  
     atm\_t \* atm,  
     obs\_t \* obs )

Determine ray paths and compute radiative transfer.

Definition at line 3033 of file [jurassic.c](#).

```

03036     {
03037
03038     int *mask;
03039
03040     /* Allocate... */
03041     ALLOC(mask, int,
03042           ND * NR);
03043
03044     /* Save observation mask... */
03045     for (int id = 0; id < ctl->nd; id++)
03046         for (int ir = 0; ir < obs->nr; ir++)
03047             mask[id * NR + ir] = !gsl_finite(obs->rad[id][ir]);
03048
03049     /* Hydrostatic equilibrium... */
03050     hydrostatic(ctl, atm);
03051
03052     /* EGA forward model... */
03053     if (ctl->formod == 1)
03054         for (int ir = 0; ir < obs->nr; ir++)
03055             formod_pencil(ctl, atm, obs, ir);
03056
03057     /* Call RFM... */
03058     else if (ctl->formod == 2)
03059         formod_rfm(ctl, atm, obs);
03060
03061     /* Apply field-of-view convolution... */
03062     formod_fov(ctl, obs);
03063
03064     /* Convert radiance to brightness temperature... */
03065     if (ctl->write_bbt)
03066         for (int id = 0; id < ctl->nd; id++)
03067             for (int ir = 0; ir < obs->nr; ir++)

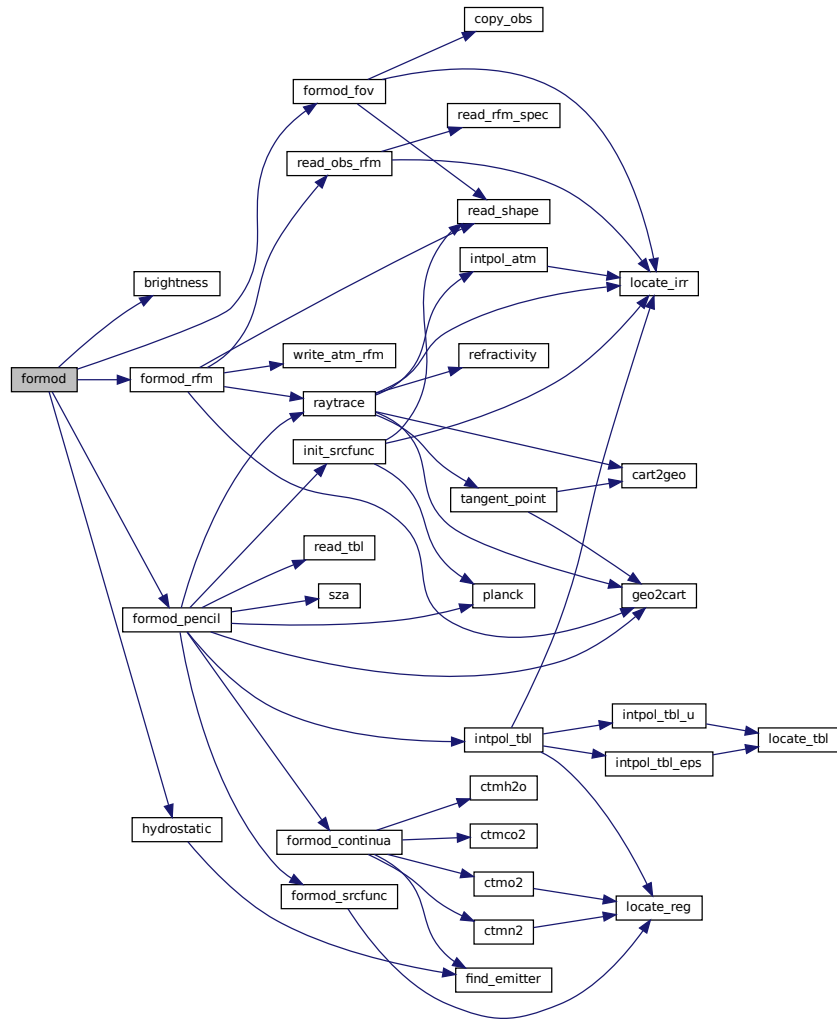
```

```

03068     obs->rad[id][ir] = brightness(obs->rad[id][ir], ctl->nu[id]);
03069
03070     /* Apply observation mask... */
03071     for (int id = 0; id < ctl->nd; id++)
03072         for (int ir = 0; ir < obs->nr; ir++)
03073             if (mask[id * NR + ir])
03074                 obs->rad[id][ir] = GSL_NAN;
03075
03076     /* Free... */
03077     free(mask);
03078 }

```

Here is the call graph for this function:



**5.17.2.14 formod\_continua()** void formod\_continua (

```

    ctl_t * ctl,
    los_t * los,
    int ip,
    double * beta )

```

Compute absorption coefficient of continua.

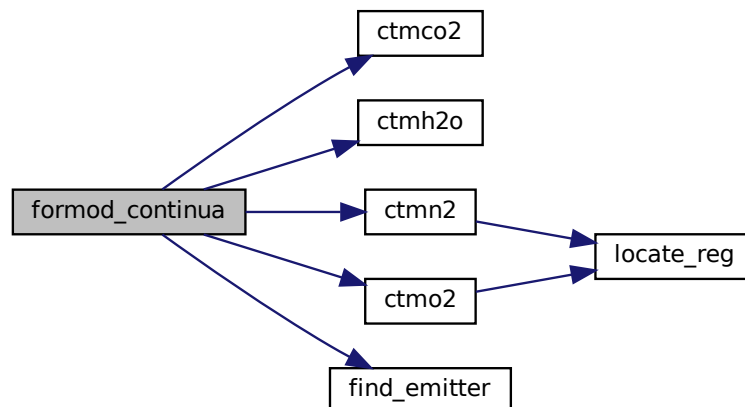
Definition at line 3082 of file [jurassic.c](#).

```

03086     {
03087
03088     static int ig_co2 = -999, ig_h2o = -999;
03089
03090     /* Extinction... */
03091     for (int id = 0; id < ctl->nd; id++)
03092         beta[id] = los->k[ip][id];
03093
03094     /* CO2 continuum... */
03095     if (ctl->ctm_co2) {
03096         if (ig_co2 == -999)
03097             ig_co2 = find_emitter(ctl, "CO2");
03098         if (ig_co2 >= 0)
03099             for (int id = 0; id < ctl->nd; id++)
03100                 beta[id] += ctmco2(ctl->nu[id], los->p[ip], los->t[ip],
03101                                     los->u[ip][ig_co2]) / los->ds[ip];
03102     }
03103
03104     /* H2O continuum... */
03105     if (ctl->ctm_h2o) {
03106         if (ig_h2o == -999)
03107             ig_h2o = find_emitter(ctl, "H2O");
03108         if (ig_h2o >= 0)
03109             for (int id = 0; id < ctl->nd; id++)
03110                 beta[id] += ctmh2o(ctl->nu[id], los->p[ip], los->t[ip],
03111                                     los->q[ip][ig_h2o], los->u[ip][ig_h2o])
03112                                     / los->ds[ip];
03113     }
03114
03115     /* N2 continuum... */
03116     if (ctl->ctm_n2)
03117         for (int id = 0; id < ctl->nd; id++)
03118             beta[id] += ctmn2(ctl->nu[id], los->p[ip], los->t[ip]);
03119
03120     /* O2 continuum... */
03121     if (ctl->ctm_o2)
03122         for (int id = 0; id < ctl->nd; id++)
03123             beta[id] += ctmo2(ctl->nu[id], los->p[ip], los->t[ip]);
03124 }

```

Here is the call graph for this function:



**5.17.2.15 formod\_fov()** void formod\_fov (  
     ctl\_t \* ctl,  
     obs\_t \* obs )

Apply field of view convolution.

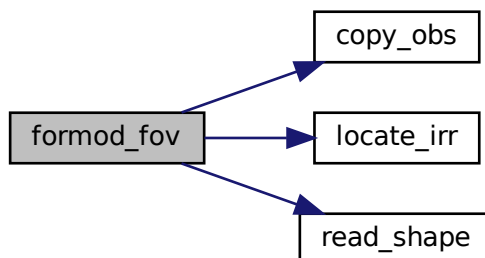
Definition at line 3128 of file [jurassic.c](#).

```

3130     {
3131
3132     static double dz[NSHAPE], w[NSHAPE];
3133
3134     static int init = 0, n;
3135
3136     obs_t *obs2;
3137
3138     double rad[ND][NR], tau[ND][NR], z[NR];
3139
3140     /* Do not take into account FOV... */
3141     if (ctl->fov[0] == '-')
3142         return;
3143
3144     /* Initialize FOV data... */
3145     if (!init) {
3146         init = 1;
3147         read_shape(ctl->fov, dz, w, &n);
3148     }
3149
3150     /* Allocate... */
3151     ALLOC(obs2, obs_t, 1);
3152
3153     /* Copy observation data... */
3154     copy_obs(ctl, obs2, obs, 0);
3155
3156     /* Loop over ray paths... */
3157     for (int ir = 0; ir < obs->nr; ir++) {
3158
3159         /* Get radiance and transmittance profiles... */
3160         int nz = 0;
3161         for (int ir2 = GSL_MAX(ir - NFOV, 0);
3162              ir2 < GSL_MIN(ir + 1 + NFOV, obs->nr); ir2++)
3163             if (obs->time[ir2] == obs->time[ir]) {
3164                 z[nz] = obs2->vpz[ir2];
3165                 for (int id = 0; id < ctl->nd; id++) {
3166                     rad[id][nz] = obs2->rad[id][ir2];
3167                     tau[id][nz] = obs2->tau[id][ir2];
3168                 }
3169                 nz++;
3170             }
3171         if (nz < 2)
3172             ERRMSG("Cannot apply FOV convolution!");
3173
3174         /* Convolute profiles with FOV... */
3175         double wsum = 0;
3176         for (int id = 0; id < ctl->nd; id++) {
3177             obs->rad[id][ir] = 0;
3178             obs->tau[id][ir] = 0;
3179         }
3180         for (int i = 0; i < n; i++) {
3181             double zfov = obs->vpz[ir] + dz[i];
3182             int idx = locate_irr(z, nz, zfov);
3183             for (int id = 0; id < ctl->nd; id++) {
3184                 obs->rad[id][ir] += w[i]
3185                     * LIN(z[idx], rad[id][idx], z[idx + 1], rad[id][idx + 1], zfov);
3186                 obs->tau[id][ir] += w[i]
3187                     * LIN(z[idx], tau[id][idx], z[idx + 1], tau[id][idx + 1], zfov);
3188             }
3189             wsum += w[i];
3190         }
3191         for (int id = 0; id < ctl->nd; id++) {
3192             obs->rad[id][ir] /= wsum;
3193             obs->tau[id][ir] /= wsum;
3194         }
3195     }
3196
3197     /* Free... */
3198     free(obs2);
3199 }

```

Here is the call graph for this function:



**5.17.2.16 formod\_pencil()** void formod\_pencil (  
     ctl\_t \* ctl,  
     atm\_t \* atm,  
     obs\_t \* obs,  
     int ir )

Compute radiative transfer for a pencil beam.

Definition at line 3203 of file jurassic.c.

```

03207     {
03208
03209     static tbl_t *tbl;
03210
03211     static int init = 0;
03212
03213     los_t *los;
03214
03215     double beta_ctm[ND], rad[ND], tau[ND], tau_refl[ND],
03216           tau_path[ND][NG], tau_gas[ND], x0[3], x1[3];
03217
03218     /* Initialize look-up tables... */
03219     if (!init) {
03220         init = 1;
03221         ALLOC(tbl, tbl_t, 1);
03222         read_tbl(ctl, tbl);
03223         init_srcfunc(ctl, tbl);
03224     }
03225
03226     /* Allocate... */
03227     ALLOC(los, los_t, 1);
03228
03229     /* Initialize... */
03230     for (int id = 0; id < ctl->nd; id++) {
03231         rad[id] = 0;
03232         tau[id] = 1;
03233         for (int ig = 0; ig < ctl->ng; ig++)
03234             tau_path[id][ig] = 1;
03235     }
03236
03237     /* Raytracing... */
03238     raytrace(ctl, atm, obs, los, ir);
03239
03240     /* Loop over LOS points... */
03241     for (int ip = 0; ip < los->np; ip++) {
03242
03243         /* Get trace gas transmittance... */
03244         intpol_tbl(ctl, tbl, los, ip, tau_path, tau_gas);
03245
03246         /* Get continuum absorption... */
  
```



```

03247     formod_continua(ctl, los, ip, beta_ctm);
03248
03249     /* Compute Planck function... */
03250     formod_srcfunc(ctl, tbl, los->t[ip], los->src[ip]);
03251
03252     /* Loop over channels... */
03253     for (int id = 0; id < ctl->nd; id++)
03254     {
03255         /* Get segment emissivity... */
03256         los->eps[ip][id] = 1 - tau_gas[id] * exp(-beta_ctm[id] * los->ds[ip]);
03257
03258         /* Compute radiance... */
03259         rad[id] += los->src[ip][id] * los->eps[ip][id] * tau[id];
03260
03261         /* Compute path transmittance... */
03262         tau[id] *= (1 - los->eps[ip][id]);
03263     }
03264 }
03265
03266 /* Check whether LOS hit the ground... */
03267 if (ctl->sftype >= 1 && los->sft > 0) {
03268
03269     /* Add surface emissions... */
03270     double src_sf[ND];
03271     formod_srcfunc(ctl, tbl, los->sft, src_sf);
03272     for (int id = 0; id < ctl->nd; id++)
03273         rad[id] += los->sfeps[id] * src_sf[id] * tau[id];
03274
03275     /* Check reflectivity... */
03276     int refl = 0;
03277     if (ctl->sftype >= 2)
03278         for (int id = 0; id < ctl->nd; id++)
03279             if (los->sfeps[id] < 1) {
03280                 refl = 1;
03281                 break;
03282             }
03283
03284     /* Calculate reflection... */
03285     if (refl) {
03286         /* Initialize... */
03287         for (int id = 0; id < ctl->nd; id++)
03288             tau_refl[id] = 1;
03289
03290         /* Add down-welling radiance... */
03291         for (int ip = los->np - 1; ip >= 0; ip--)
03292             for (int id = 0; id < ctl->nd; id++) {
03293                 rad[id] += los->src[ip][id] * los->eps[ip][id] * tau_refl[id]
03294                     * tau[id] * (1 - los->sfeps[id]);
03295                 tau_refl[id] *= (1 - los->eps[ip][id]);
03296             }
03297
03298         /* Add solar term... */
03299         if (ctl->sftype >= 3) {
03300             /* Get solar zenith angle... */
03301             double sza2;
03302             if (ctl->sfsza < 0)
03303                 sza2 =
03304                     sza(obs->time[ir], los->lon[los->np - 1], los->lat[los->np - 1]);
03305             else
03306                 sza2 = ctl->sfsza;
03307
03308             /* Check solar zenith angle... */
03309             if (sza2 < 89.999) {
03310                 /* Get angle of incidence... */
03311                 geo2cart(los->z[los->np - 1], los->lon[los->np - 1],
03312                     los->lat[los->np - 1], x0);
03313                 geo2cart(los->z[0], los->lon[0], los->lat[0], x1);
03314                 for (int i = 0; i < 3; i++)
03315                     x1[i] -= x0[i];
03316                 double cosa = DOTP(x0, x1) / NORM(x0) / NORM(x1);
03317
03318                 /* Get ratio of SZA and incident radiation... */
03319                 double rcos = cosa / cos(sza2 * M_PI / 180.);
03320
03321                 /* Add solar radiation... */
03322                 for (int id = 0; id < ctl->nd; id++)
03323                     rad[id] += 6.764e-5 / (2. * M_PI) * planck(TSUN, ctl->nu[id])
03324                         * tau_refl[id] * (1 - los->sfeps[id]) * tau[id] * rcos;
03325             }
03326         }
03327     }
03328 }
03329
03330 }
03331
03332 }
03333

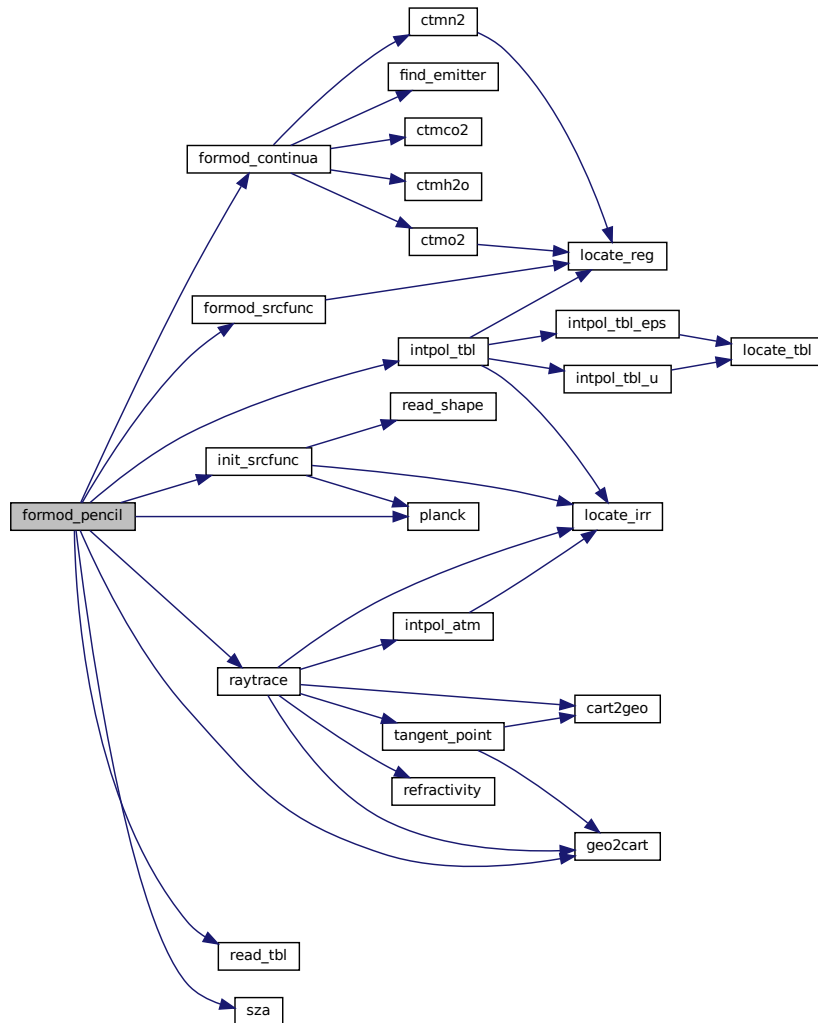
```

```

03334  /* Copy results... */
03335  for (int id = 0; id < ctl->nd; id++) {
03336      obs->rad[id][ir] = rad[id];
03337      obs->tau[id][ir] = tau[id];
03338  }
03339
03340  /* Free... */
03341  free(los);
03342  }

```

Here is the call graph for this function:



**5.17.2.17 formod\_rfm()** void formod\_rfm (  
     ctl\_t \* ctl,  
     atm\_t \* atm,  
     obs\_t \* obs )

Apply RFM for radiative transfer calculations.

Definition at line 3346 of file [jurassic.c](#).

```

03349         {
03350
03351     los_t *los;
03352
03353     FILE *out;
03354
03355     char cmd[2 * LEN], filename[2 * LEN],
03356         rfmflg[LEN] = { "RAD TRA MIX LIN SFC" };
03357
03358     double f[NSHAPE], nu[NSHAPE], nu0, nu1, obsz = -999, tsurf,
03359         xd[3], xo[3], xv[3], z[NR], zmin, zmax;
03360
03361     int i, id, ig, ip, ir, iw, n, nadir = 0;
03362
03363     /* Allocate... */
03364     ALLOC(los, los_t, 1);
03365
03366     /* Check observer positions... */
03367     for (ir = 1; ir < obs->nr; ir++)
03368         if (obs->obsz[ir] != obs->obsz[0]
03369             || obs->obslon[ir] != obs->obslon[0]
03370             || obs->obslat[ir] != obs->obslat[0])
03371             ERRMSG("RFM interface requires identical observer positions!");
03372
03373     /* Check extinction data... */
03374     for (iw = 0; iw < ctl->nw; iw++)
03375         for (ip = 0; ip < atm->np; ip++)
03376             if (atm->k[iw][ip] != 0)
03377                 ERRMSG("RFM interface cannot handle extinction data!");
03378
03379     /* Get altitude range of atmospheric data... */
03380     gsl_stats_minmax(&zmin, &zmax, atm->z, 1, (size_t) atm->np);
03381
03382     /* Observer within atmosphere? */
03383     if (obs->obsz[0] >= zmin && obs->obsz[0] <= zmax) {
03384         obsz = obs->obsz[0];
03385         strcat(rfmflg, " OBS");
03386     }
03387
03388     /* Determine tangent altitude or air mass factor... */
03389     for (ir = 0; ir < obs->nr; ir++) {
03390
03391         /* Raytracing... */
03392         raytrace(ctl, atm, obs, los, ir);
03393
03394         /* Nadir? */
03395         if (obs->tpz[ir] <= zmin) {
03396             geo2cart(obs->obsz[ir], obs->obslon[ir], obs->obslat[ir], xo);
03397             geo2cart(obs->vpz[ir], obs->vpplon[ir], obs->vpplat[ir], xv);
03398             for (i = 0; i < 3; i++)
03399                 xd[i] = xo[i] - xv[i];
03400             z[ir] = NORM(xo) * NORM(xd) / DOTP(xo, xd);
03401             nadir++;
03402         } else
03403             z[ir] = obs->tpz[ir];
03404     }
03405     if (nadir > 0 && nadir < obs->nr)
03406         ERRMSG("Limb and nadir not simultaneously possible!");
03407
03408     /* Nadir? */
03409     if (nadir)
03410         strcat(rfmflg, " NAD");
03411
03412     /* Get surface temperature... */
03413     tsurf = atm->t[gsl_stats_min_index(atm->z, 1, (size_t) atm->np)];
03414
03415     /* Refraction? */
03416     if (!nadir && !ctl->refrac)
03417         strcat(rfmflg, " GEO");
03418
03419     /* Continua? */
03420     if (ctl->ctm_co2 || ctl->ctm_h2o || ctl->ctm_n2 || ctl->ctm_o2)
03421         strcat(rfmflg, " CTM");
03422
03423     /* Write atmospheric data file... */
03424     write_atm_rfm("rfm.atm", ctl, atm);
03425
03426     /* Loop over channels... */
03427     for (id = 0; id < ctl->nd; id++) {
03428
03429         /* Read filter function... */
03430         sprintf(filename, "%s_%.4f.filt", ctl->tblbase, ctl->nu[id]);
03431         read_shape(filename, nu, f, &n);
03432
03433         /* Set spectral range... */
03434         nu0 = nu[0];
03435         nu1 = nu[n - 1];

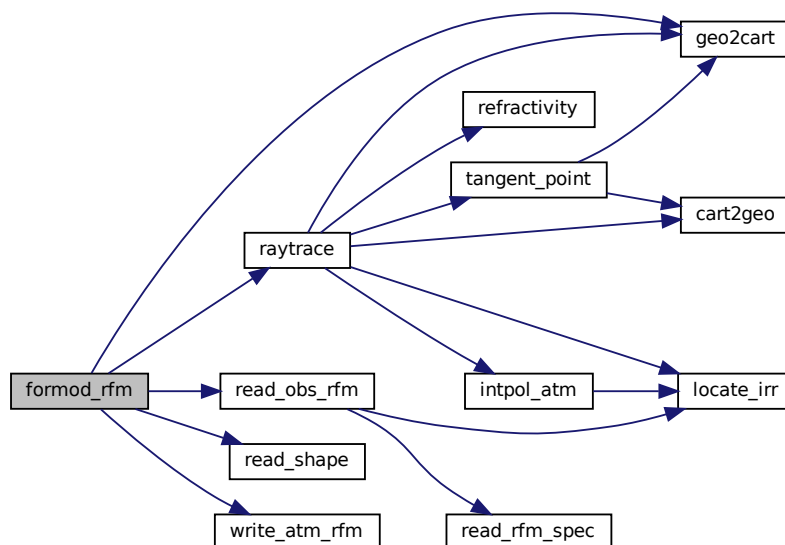
```

```

03436
03437  /* Create RFM driver file... */
03438  if (! (out = fopen("rfm.drv", "w")))
03439      ERRMSG("Cannot create file!");
03440  fprintf(out, "HDR\nRFM call by JURASSIC.\n");
03441  fprintf(out, "FLG\n%s\n", rfmflg);
03442  fprintf(out, "SPC\n%.4f %.4f 0.0005\n", nu0, nul);
03443  fprintf(out, "GAS\n");
03444  for (ig = 0; ig < ctl->ng; ig++)
03445      fprintf(out, "s\n", ctl->emitter[ig]);
03446  fprintf(out, "ATM\nrfm.atm\n");
03447  fprintf(out, "TAN\n");
03448  for (ir = 0; ir < obs->nr; ir++)
03449      fprintf(out, "g\n", z[ir]);
03450  fprintf(out, "SFC\n%g 1.0\n", tsurf);
03451  if (obsz >= 0)
03452      fprintf(out, "OBS\n%g\n", obsz);
03453  fprintf(out, "HIT\n%s\n", ctl->rfmhit);
03454  fprintf(out, "XSC\n");
03455  for (ig = 0; ig < ctl->ng; ig++)
03456      if (ctl->rfmxsc[ig][0] != '-')
03457          fprintf(out, "s\n", ctl->rfmxsc[ig]);
03458  fprintf(out, "END\n");
03459  fclose(out);
03460
03461  /* Remove temporary files... */
03462  if (system("rm -f rfm.runlog rad_*.asc tra_*.asc"))
03463      ERRMSG("Cannot remove temporary files!");
03464
03465  /* Call RFM... */
03466  sprintf(cmd, "echo | %s", ctl->rfmbin);
03467  if (system(cmd))
03468      ERRMSG("Error while calling RFM!");
03469
03470  /* Read data... */
03471  for (ir = 0; ir < obs->nr; ir++) {
03472      obs->rad[id][ir] = read_obs_rfm("rad", z[ir], nu, f, n) * 1e-5;
03473      obs->tau[id][ir] = read_obs_rfm("tra", z[ir], nu, f, n);
03474  }
03475  }
03476
03477  /* Remove temporary files... */
03478  if (system("rm -f rfm.drv rfm.atm rfm.runlog rad_*.asc tra_*.asc"))
03479      ERRMSG("Error while removing temporary files!");
03480
03481  /* Free... */
03482  free(los);
03483 }

```

Here is the call graph for this function:



**5.17.2.18 formod\_srcfunc()** void formod\_srcfunc (  
    ctl\_t \* ctl,  
    tbl\_t \* tbl,  
    double t,  
    double \* src )

Compute Planck source function.

Definition at line 3487 of file [jurassic.c](#).

```
03491     {  
03492  
03493     /* Determine index in temperature array... */  
03494     int it = locate_reg(tbl->st, TBLNS, t);  
03495  
03496     /* Interpolate Planck function value... */  
03497     for (int id = 0; id < ctl->nd; id++)  
03498         src[id] = LIN(tbl->st[it], tbl->sr[it][id],  
03499                     tbl->st[it + 1], tbl->sr[it + 1][id], t);  
03500 }
```

Here is the call graph for this function:



**5.17.2.19 geo2cart()** void geo2cart (  
    double z,  
    double lon,  
    double lat,  
    double \* x )

Convert geolocation to Cartesian coordinates.

Definition at line 3504 of file [jurassic.c](#).

```
03508     {  
03509  
03510     double radius = z + RE;  
03511  
03512     x[0] = radius * cos(lat / 180 * M_PI) * cos(lon / 180 * M_PI);  
03513     x[1] = radius * cos(lat / 180 * M_PI) * sin(lon / 180 * M_PI);  
03514     x[2] = radius * sin(lat / 180 * M_PI);  
03515 }
```

**5.17.2.20 hydrostatic()** void hydrostatic (  
     ctl\_t \* ctl,  
     atm\_t \* atm )

Set hydrostatic equilibrium.

Definition at line 3519 of file [jurassic.c](#).

```

03521     {
03522
03523         const double mmair = 28.96456e-3, mmh2o = 18.0153e-3;
03524
03525         const int ipts = 20;
03526
03527         static int ig_h2o = -999;
03528
03529         double dzmin = 1e99, e = 0;
03530
03531         int ipref = 0;
03532
03533         /* Check reference height... */
03534         if (ctl->hydZ < 0)
03535             return;
03536
03537         /* Determine emitter index of H2O... */
03538         if (ig_h2o == -999)
03539             ig_h2o = find_emitter(ctl, "H2O");
03540
03541         /* Find air parcel next to reference height... */
03542         for (int ip = 0; ip < atm->np; ip++)
03543             if (fabs(atm->z[ip] - ctl->hydZ) < dzmin) {
03544                 dzmin = fabs(atm->z[ip] - ctl->hydZ);
03545                 ipref = ip;
03546             }
03547
03548         /* Upper part of profile... */
03549         for (int ip = ipref + 1; ip < atm->np; ip++) {
03550             double mean = 0;
03551             for (int i = 0; i < ipts; i++) {
03552                 if (ig_h2o >= 0)
03553                     e = LIN(0.0, atm->q[ig_h2o][ip - 1],
03554                             ipts - 1.0, atm->q[ig_h2o][ip], (double) i);
03555                 mean += (e * mmh2o + (1 - e) * mmair)
03556                     * G0 / RI
03557                     / LIN(0.0, atm->t[ip - 1], ipts - 1.0, atm->t[ip], (double) i) / ipts;
03558             }
03559
03560             /* Compute p(z,T)... */
03561             atm->p[ip] =
03562                 exp(log(atm->p[ip - 1]) - mean * 1000 * (atm->z[ip] - atm->z[ip - 1]));
03563         }
03564
03565         /* Lower part of profile... */
03566         for (int ip = ipref - 1; ip >= 0; ip--) {
03567             double mean = 0;
03568             for (int i = 0; i < ipts; i++) {
03569                 if (ig_h2o >= 0)
03570                     e = LIN(0.0, atm->q[ig_h2o][ip + 1],
03571                             ipts - 1.0, atm->q[ig_h2o][ip], (double) i);
03572                 mean += (e * mmh2o + (1 - e) * mmair)
03573                     * G0 / RI
03574                     / LIN(0.0, atm->t[ip + 1], ipts - 1.0, atm->t[ip], (double) i) / ipts;
03575             }
03576
03577             /* Compute p(z,T)... */
03578             atm->p[ip] =
03579                 exp(log(atm->p[ip + 1]) - mean * 1000 * (atm->z[ip] - atm->z[ip + 1]));
03580         }
03581     }

```

Here is the call graph for this function:



**5.17.2.21 idx2name()** void idx2name (  
     ctl\_t \* ctl,  
     int idx,  
     char \* quantity )

Determine name of state vector quantity for given index.

Definition at line 3585 of file [jurassic.c](#).

```

03588     {
03589
03590     if (idx == IDXP)
03591         sprintf(quantity, "PRESSURE");
03592
03593     if (idx == IDXT)
03594         sprintf(quantity, "TEMPERATURE");
03595
03596     for (int ig = 0; ig < ctl->ng; ig++)
03597         if (idx == IDXQ(ig))
03598             sprintf(quantity, "%s", ctl->emitter[ig]);
03599
03600     for (int iw = 0; iw < ctl->nw; iw++)
03601         if (idx == IDXK(iw))
03602             sprintf(quantity, "EXTINCT_WINDOW_%d", iw);
03603
03604     if (idx == IDXCLZ)
03605         sprintf(quantity, "CLOUD_HEIGHT");
03606
03607     if (idx == IDXCLDZ)
03608         sprintf(quantity, "CLOUD_DEPTH");
03609
03610     for (int icl = 0; icl < ctl->ncl; icl++)
03611         if (idx == IDXCLK(icl))
03612             sprintf(quantity, "CLOUD_EXTINCT_%.4f", ctl->clnu[icl]);
03613
03614     if (idx == IDXSFZ)
03615         sprintf(quantity, "SURFACE_HEIGHT");
03616
03617     if (idx == IDXSFP)
03618         sprintf(quantity, "SURFACE_PRESSURE");
03619
03620     if (idx == IDXSFT)
03621         sprintf(quantity, "SURFACE_TEMPERATURE");
03622
03623     for (int isf = 0; isf < ctl->nsf; isf++)
03624         if (idx == IDXSFEP(isf))
03625             sprintf(quantity, "SURFACE_EMISSIVITY_%.4f", ctl->sfnu[isf]);
03626 }

```

**5.17.2.22 init\_srcfunc()** void init\_srcfunc (  
     ctl\_t \* ctl,  
     tbl\_t \* tbl )

Initialize source function table.

Definition at line 3630 of file [jurassic.c](#).

```

03632     {
03633
03634     char filename[2 * LEN];
03635
03636     double f[NSHAPE], nu[NSHAPE];
03637
03638     int n;
03639
03640     /* Write info... */
03641     LOG(1, "Initialize source function table...");
03642
03643     /* Loop over channels... */
03644     for (int id = 0; id < ctl->nd; id++) {
03645
03646         /* Read filter function... */

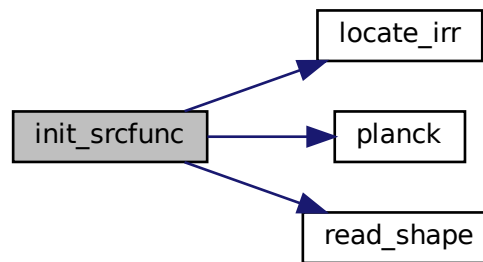
```

```

03647     sprintf(filename, "%s_%.4f.filt", ctl->tblbase, ctl->nu[id]);
03648     read_shape(filename, nu, f, &n);
03649
03650     /* Get minimum grid spacing... */
03651     double dnu = 1.0;
03652     for (int i = 1; i < n; i++)
03653         dnu = GSL_MIN(dnu, nu[i] - nu[i - 1]);
03654
03655     /* Compute source function table... */
03656     #pragma omp parallel for default(none) shared(ctl,tbl,id,nu,f,n,dnu)
03657     for (int it = 0; it < TBLNS; it++) {
03658
03659         /* Set temperature... */
03660         tbl->st[it] = LIN(0.0, TMIN, TBLNS - 1.0, TMAX, (double) it);
03661
03662         /* Integrate Planck function... */
03663         double fsum = tbl->sr[it][id] = 0;
03664         for (double fnu = nu[0]; fnu <= nu[n - 1]; fnu += dnu) {
03665             int i = locate_irr(nu, n, fnu);
03666             double ff = LIN(nu[i], f[i], nu[i + 1], f[i + 1], fnu);
03667             fsum += ff;
03668             tbl->sr[it][id] += ff * planck(tbl->st[it], fnu);
03669         }
03670         tbl->sr[it][id] /= fsum;
03671     }
03672 }
03673 }

```

Here is the call graph for this function:



**5.17.2.23 intpol\_atm()** void intpol\_atm (

```

    ctl_t * ctl,
    atm_t * atm,
    double z,
    double * p,
    double * t,
    double * q,
    double * k )

```

Interpolate atmospheric data.

Definition at line 3677 of file [jurassic.c](#).

```

03684     {
03685
03686         /* Get array index... */
03687         int ip = locate_irr(atm->z, atm->np, z);
03688
03689         /* Interpolate... */
03690         *p = EXP(atm->z[ip], atm->p[ip], atm->z[ip + 1], atm->p[ip + 1], z);

```



```

03691  *t = LIN(atm->z[ip], atm->t[ip], atm->z[ip + 1], atm->t[ip + 1], z);
03692  for (int ig = 0; ig < ctl->ng; ig++)
03693      q[ig] =
03694          LIN(atm->z[ip], atm->q[ig][ip], atm->z[ip + 1], atm->q[ig][ip + 1], z);
03695  for (int iw = 0; iw < ctl->nw; iw++)
03696      k[iw] =
03697          LIN(atm->z[ip], atm->k[iw][ip], atm->z[ip + 1], atm->k[iw][ip + 1], z);
03698  }

```

Here is the call graph for this function:



**5.17.2.24 intpol\_tbl()** void intpol\_tbl (

```

    ctl_t * ctl,
    tbl_t * tbl,
    los_t * los,
    int ip,
    double tau_path[ND][NG],
    double tau_seg[ND] )

```

Get transmittance from look-up tables.

Definition at line 3702 of file [jurassic.c](#).

```

03708  {
03709
03710      double eps, u;
03711
03712      /* Loop over channels... */
03713      for (int id = 0; id < ctl->nd; id++) {
03714
03715          /* Initialize... */
03716          tau_seg[id] = 1;
03717
03718          /* Loop over emitters.... */
03719          for (int ig = 0; ig < ctl->ng; ig++) {
03720
03721              /* Check size of table (pressure)... */
03722              if (tbl->np[id][ig] < 30)
03723                  eps = 0;
03724
03725              /* Check transmittance... */
03726              else if (tau_path[id][ig] < 1e-9)
03727                  eps = 1;
03728
03729              /* Interpolate... */
03730              else {
03731
03732                  /* Determine pressure and temperature indices... */
03733                  int ipr = locate_irr(tbl->p[id][ig], tbl->np[id][ig], los->p[ip]);
03734                  int it0 =
03735                      locate_reg(tbl->t[id][ig][ipr], tbl->nt[id][ig][ipr], los->t[ip]);
03736                  int it1 =
03737                      locate_reg(tbl->t[id][ig][ipr + 1], tbl->nt[id][ig][ipr + 1],
03738                                los->t[ip]);
03739
03740                  /* Check size of table (temperature and column density)... */
03741                  if (tbl->nt[id][ig][ipr] < 2 || tbl->nt[id][ig][ipr + 1] < 2
03742                      || tbl->nu[id][ig][ipr][it0] < 2
03743                      || tbl->nu[id][ig][ipr][it0 + 1] < 2
03744                      || tbl->nu[id][ig][ipr + 1][it1] < 2

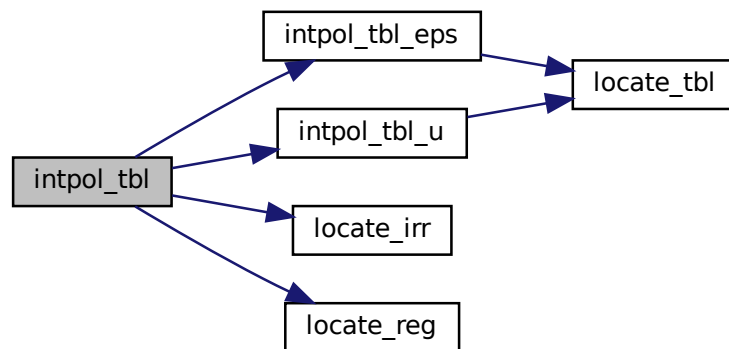
```

```

03745     || tbl->nu[id][ig][ipr + 1][it1 + 1] < 2)
03746     eps = 0;
03747
03748     else {
03749
03750         /* Get emissivities of extended path... */
03751         u = intpol_tbl_u(tbl, ig, id, ipr, it0, 1 - tau_path[id][ig]);
03752         double eps00
03753             = intpol_tbl_eps(tbl, ig, id, ipr, it0, u + los->u[ip][ig]);
03754
03755         u = intpol_tbl_u(tbl, ig, id, ipr, it0 + 1, 1 - tau_path[id][ig]);
03756         double eps01 =
03757             intpol_tbl_eps(tbl, ig, id, ipr, it0 + 1, u + los->u[ip][ig]);
03758
03759         u = intpol_tbl_u(tbl, ig, id, ipr + 1, it1, 1 - tau_path[id][ig]);
03760         double eps10 =
03761             intpol_tbl_eps(tbl, ig, id, ipr + 1, it1, u + los->u[ip][ig]);
03762
03763         u =
03764             intpol_tbl_u(tbl, ig, id, ipr + 1, it1 + 1, 1 - tau_path[id][ig]);
03765         double eps11 =
03766             intpol_tbl_eps(tbl, ig, id, ipr + 1, it1 + 1, u + los->u[ip][ig]);
03767
03768         /* Interpolate with respect to temperature... */
03769         eps00 = LIN(tbl->t[id][ig][ipr][it0], eps00,
03770             tbl->t[id][ig][ipr][it0 + 1], eps01, los->t[ip]);
03771         eps11 = LIN(tbl->t[id][ig][ipr + 1][it1], eps10,
03772             tbl->t[id][ig][ipr + 1][it1 + 1], eps11, los->t[ip]);
03773
03774         /* Interpolate with respect to pressure... */
03775         eps00 = LIN(tbl->p[id][ig][ipr], eps00,
03776             tbl->p[id][ig][ipr + 1], eps11, los->p[ip]);
03777
03778         /* Check emssivity range... */
03779         eps00 = GSL_MAX(GSL_MIN(eps00, 1), 0);
03780
03781         /* Determine segment emissivity... */
03782         eps = 1 - (1 - eps00) / tau_path[id][ig];
03783     }
03784 }
03785
03786 /* Get transmittance of extended path... */
03787 tau_path[id][ig] *= (1 - eps);
03788
03789 /* Get segment transmittance... */
03790 tau_seg[id] *= (1 - eps);
03791 }
03792 }
03793 }

```

Here is the call graph for this function:



```

5.17.2.25 intpol_tbl_eps() double intpol_tbl_eps (
    tbl_t * tbl,
    int ig,
    int id,
    int ip,
    int it,
    double u )

```

Interpolate emissivity from look-up tables.

Definition at line 3797 of file [jurassic.c](#).

```

03803     {
03804
03805     /* Lower boundary... */
03806     if (u < tbl->u[id][ig][ip][it][0])
03807         return LIN(0, 0, tbl->u[id][ig][ip][it][0], tbl->eps[id][ig][ip][it][0],
03808             u);
03809
03810     /* Upper boundary... */
03811     else if (u > tbl->u[id][ig][ip][it][tbl->nu[id][ig][ip][it] - 1])
03812         return LIN(tbl->u[id][ig][ip][it][tbl->nu[id][ig][ip][it] - 1],
03813             tbl->eps[id][ig][ip][it][tbl->nu[id][ig][ip][it] - 1],
03814             1e30, 1, u);
03815
03816     /* Interpolation... */
03817     else {
03818
03819         /* Get index... */
03820         int idx = locate_tbl(tbl->u[id][ig][ip][it], tbl->nu[id][ig][ip][it], u);
03821
03822         /* Interpolate... */
03823         return
03824             LIN(tbl->u[id][ig][ip][it][idx], tbl->eps[id][ig][ip][it][idx],
03825                 tbl->u[id][ig][ip][it][idx + 1], tbl->eps[id][ig][ip][it][idx + 1],
03826                 u);
03827     }
03828 }

```

Here is the call graph for this function:



```

5.17.2.26 intpol_tbl_u() double intpol_tbl_u (
    tbl_t * tbl,
    int ig,
    int id,
    int ip,
    int it,
    double eps )

```

Interpolate column density from look-up tables.

Definition at line 3832 of file [jurassic.c](#).

```

03838     {
03839
03840     /* Lower boundary... */

```

```

03841  if (eps < tbl->eps[id][ig][ip][it][0])
03842      return LIN(0, 0, tbl->eps[id][ig][ip][it][0], tbl->u[id][ig][ip][it][0],
03843                eps);
03844
03845  /* Upper boundary... */
03846  else if (eps > tbl->eps[id][ig][ip][it][tbl->nu[id][ig][ip][it] - 1])
03847      return LIN(tbl->eps[id][ig][ip][it][tbl->nu[id][ig][ip][it] - 1],
03848                tbl->u[id][ig][ip][it][tbl->nu[id][ig][ip][it] - 1],
03849                1, 1e30, eps);
03850
03851  /* Interpolation... */
03852  else {
03853
03854      /* Get index... */
03855      int idx
03856          = locate_tbl(tbl->eps[id][ig][ip][it], tbl->nu[id][ig][ip][it], eps);
03857
03858      /* Interpolate... */
03859      return
03860          LIN(tbl->eps[id][ig][ip][it][idx], tbl->u[id][ig][ip][it][idx],
03861            tbl->eps[id][ig][ip][it][idx + 1], tbl->u[id][ig][ip][it][idx + 1],
03862            eps);
03863  }
03864 }

```

Here is the call graph for this function:



#### 5.17.2.27 jsec2time() void jsec2time (

```

    double jsec,
    int * year,
    int * mon,
    int * day,
    int * hour,
    int * min,
    int * sec,
    double * remain )

```

Convert seconds to date.

Definition at line 3868 of file [jurassic.c](#).

```

03876  {
03877
03878      struct tm t0, *t1;
03879
03880      t0.tm_year = 100;
03881      t0.tm_mon = 0;
03882      t0.tm_mday = 1;
03883      t0.tm_hour = 0;
03884      t0.tm_min = 0;
03885      t0.tm_sec = 0;
03886
03887      time_t jsec0 = (time_t) jsec + timegm(&t0);
03888      t1 = gmtime(&jsec0);
03889
03890      *year = t1->tm_year + 1900;
03891      *mon = t1->tm_mon + 1;
03892      *day = t1->tm_mday;
03893      *hour = t1->tm_hour;

```

```

03894  *min = t1->tm_min;
03895  *sec = t1->tm_sec;
03896  *remain = jsec - floor(jsec);
03897 }

```

**5.17.2.28 kernel()** void kernel (

```

    ctl_t * ctl,
    atm_t * atm,
    obs_t * obs,
    gsl_matrix * k )

```

Compute Jacobians.

Definition at line 3901 of file [jurassic.c](#).

```

03905  {
03906
03907  atm_t *atml;
03908  obs_t *obs1;
03909
03910  gsl_vector *x0, *x1, *yy0, *yy1;
03911
03912  int *iqa;
03913
03914  /* Get sizes... */
03915  size_t m = k->size1;
03916  size_t n = k->size2;
03917
03918  /* Allocate... */
03919  x0 = gsl_vector_alloc(n);
03920  yy0 = gsl_vector_alloc(m);
03921  ALLOC(iqa, int,
03922        N);
03923
03924  /* Compute radiance for undisturbed atmospheric data... */
03925  formod(ctl, atm, obs);
03926
03927  /* Compose vectors... */
03928  atm2x(ctl, atm, x0, iqa, NULL);
03929  obs2y(ctl, obs, yy0, NULL, NULL);
03930
03931  /* Initialize kernel matrix... */
03932  gsl_matrix_set_zero(k);
03933
03934  /* Loop over state vector elements... */
03935  #pragma omp parallel for default(none) shared(ctl,atm,obs,k,x0,yy0,n,m,iqa) private(x1, yy1, atml,
03936  obs1)
03937  for (size_t j = 0; j < n; j++) {
03938
03939  /* Allocate... */
03940  x1 = gsl_vector_alloc(n);
03941  yy1 = gsl_vector_alloc(m);
03942  ALLOC(atml, atm_t, 1);
03943  ALLOC(obs1, obs_t, 1);
03944
03945  /* Set perturbation size... */
03946  double h;
03947  if (iqa[j] == IDXP)
03948    h = GSL_MAX(fabs(0.01 * gsl_vector_get(x0, j)), 1e-7);
03949  else if (iqa[j] == IDXT)
03950    h = 1.0;
03951  else if (iqa[j] >= IDXQ(0) && iqa[j] < IDXQ(ctl->ng))
03952    h = GSL_MAX(fabs(0.01 * gsl_vector_get(x0, j)), 1e-15);
03953  else if (iqa[j] >= IDXK(0) && iqa[j] < IDXK(ctl->nw))
03954    h = 1e-4;
03955  else if (iqa[j] == IDXCLZ || iqa[j] == IDXCLDZ)
03956    h = 1.0;
03957  else if (iqa[j] >= IDXCLK(0) && iqa[j] < IDXCLK(ctl->ncl))
03958    h = 1e-4;
03959  else if (iqa[j] == IDXSFZ)
03960    h = 0.1;
03961  else if (iqa[j] == IDXSFP)
03962    h = 10.0;
03963  else if (iqa[j] == IDXSFT)
03964    h = 1.0;
03965  else if (iqa[j] >= IDXSFEPS(0) && iqa[j] < IDXSFEPS(ctl->nsf))
03966    h = 1e-2;
03967  else

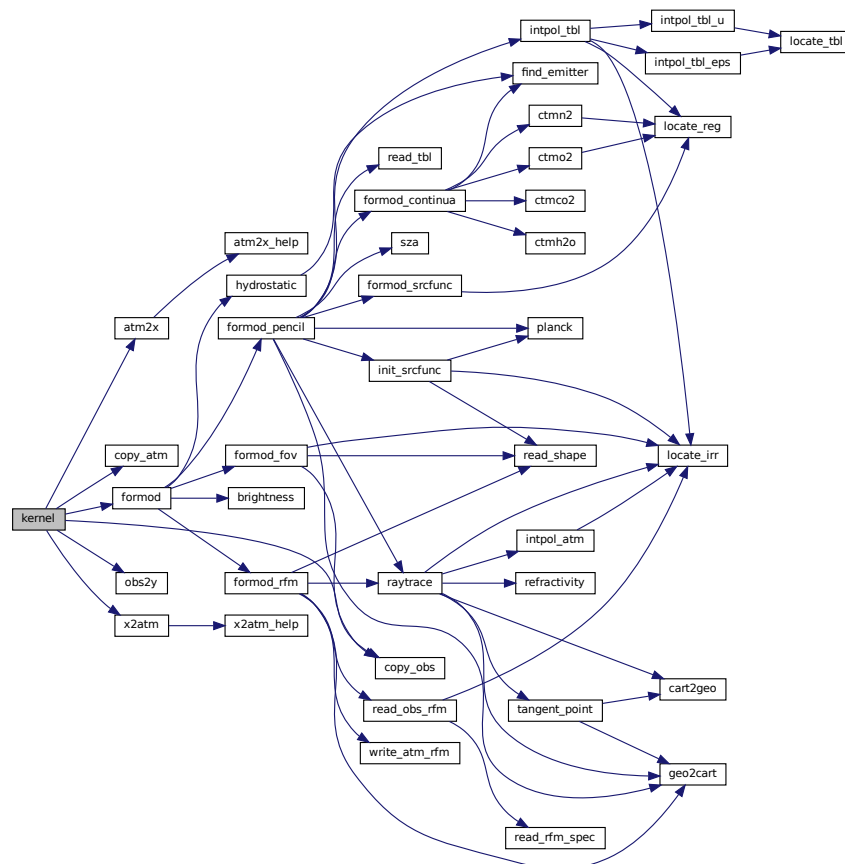
```

```

03967     ERRMSG("Cannot set perturbation size!");
03968
03969     /* Disturb state vector element... */
03970     gsl_vector_memcpy(x1, x0);
03971     gsl_vector_set(x1, j, gsl_vector_get(x1, j) + h);
03972     copy_atm(ct1, atm1, atm, 0);
03973     copy_obs(ct1, obs1, obs, 0);
03974     x2atm(ct1, x1, atm1);
03975
03976     /* Compute radiance for disturbed atmospheric data... */
03977     formod(ct1, atm1, obs1);
03978
03979     /* Compose measurement vector for disturbed radiance data... */
03980     obs2y(ct1, obs1, yy1, NULL, NULL);
03981
03982     /* Compute derivatives... */
03983     for (size_t i = 0; i < m; i++)
03984         gsl_matrix_set(k, i, j,
03985             (gsl_vector_get(yy1, i) - gsl_vector_get(yy0, i)) / h);
03986
03987     /* Free... */
03988     gsl_vector_free(x1);
03989     gsl_vector_free(yy1);
03990     free(atm1);
03991     free(obs1);
03992 }
03993
03994 /* Free... */
03995 gsl_vector_free(x0);
03996 gsl_vector_free(yy0);
03997 free(iga);
03998 }

```

Here is the call graph for this function:



**5.17.2.29 locate\_irr()** int locate\_irr (  
double \* xx,  
int n,  
double x )

Find array index for irregular grid.

Definition at line [4002](#) of file [jurassic.c](#).

```
04005     {  
04006  
04007     int ilo = 0;  
04008     int ihi = n - 1;  
04009     int i = (ihi + ilo) » 1;  
04010  
04011     if (xx[i] < xx[i + 1])  
04012         while (ihi > ilo + 1) {  
04013         i = (ihi + ilo) » 1;  
04014         if (xx[i] > x)  
04015             ihi = i;  
04016         else  
04017             ilo = i;  
04018     } else  
04019         while (ihi > ilo + 1) {  
04020         i = (ihi + ilo) » 1;  
04021         if (xx[i] <= x)  
04022             ihi = i;  
04023         else  
04024             ilo = i;  
04025     }  
04026  
04027     return ilo;  
04028 }
```

**5.17.2.30 locate\_reg()** int locate\_reg (  
double \* xx,  
int n,  
double x )

Find array index for regular grid.

Definition at line [4032](#) of file [jurassic.c](#).

```
04035     {  
04036  
04037     /* Calculate index... */  
04038     int i = (int) ((x - xx[0]) / (xx[1] - xx[0]));  
04039  
04040     /* Check range... */  
04041     if (i < 0)  
04042         return 0;  
04043     else if (i > n - 2)  
04044         return n - 2;  
04045     else  
04046         return i;  
04047 }
```

**5.17.2.31 locate\_tbl()** int locate\_tbl (  
float \* xx,  
int n,  
double x )

Find array index in float array.

Definition at line [4051](#) of file [jurassic.c](#).

```
04054     {  
04055  
04056     int ilo = 0;
```

```

04057     int ihi = n - 1;
04058     int i = (ihi + ilo) » 1;
04059
04060     while (ihi > ilo + 1) {
04061         i = (ihi + ilo) » 1;
04062         if (xx[i] > x)
04063             ihi = i;
04064         else
04065             ilo = i;
04066     }
04067
04068     return ilo;
04069 }

```

**5.17.2.32 obs2y()** size\_t obs2y (  
     ctl\_t \* ctl,  
     obs\_t \* obs,  
     gsl\_vector \* y,  
     int \* ida,  
     int \* ira )

Compose measurement vector.

Definition at line 4073 of file [jurassic.c](#).

```

04078     {
04079
04080         size_t m = 0;
04081
04082         /* Determine measurement vector... */
04083         for (int ir = 0; ir < obs->nr; ir++)
04084             for (int id = 0; id < ctl->nd; id++)
04085                 if (gsl_finite(obs->rad[id][ir])) {
04086                     if (y != NULL)
04087                         gsl_vector_set(y, m, obs->rad[id][ir]);
04088                     if (ida != NULL)
04089                         ida[m] = id;
04090                     if (ira != NULL)
04091                         ira[m] = ir;
04092                     m++;
04093                 }
04094
04095         return m;
04096 }

```

**5.17.2.33 planck()** double planck (  
     double t,  
     double nu )

Compute Planck function.

Definition at line 4100 of file [jurassic.c](#).

```

04102     {
04103
04104         return C1 * POW3(nu) / gsl_expml(C2 * nu / t);
04105     }

```



```

5.17.2.34 raytrace() void raytrace (
    ctl_t * ctl,
    atm_t * atm,
    obs_t * obs,
    los_t * los,
    int ir )

```

Do ray-tracing to determine LOS.

Definition at line 4109 of file [jurassic.c](#).

```

41114     {
41115
41116     const double h = 0.02, zrefrac = 60;
41117
41118     double ds, ex0[3], ex1[3], k[NW], lat, lon, n, ng[3], norm,
41119         p, q[NG], t, x[3], xh[3], xobs[3], xvp[3], z = 1e99, zmax, zmin;
41120
41121     int stop = 0;
41122
41123     /* Initialize... */
41124     los->np = 0;
41125     los->sft = -999;
41126     obs->tpz[ir] = obs->vpz[ir];
41127     obs->tplon[ir] = obs->vplon[ir];
41128     obs->tplat[ir] = obs->vplat[ir];
41129
41130     /* Get altitude range of atmospheric data... */
41131     gsl_stats_minmax(&zmin, &zmax, atm->z, 1, (size_t) atm->np);
41132     if (ctl->nsf > 0) {
41133         zmin = GSL_MAX(atm->sfz, zmin);
41134         if (atm->sfp > 0) {
41135             int ip = locate_irr(atm->p, atm->np, atm->sfp);
41136             double zip = LIN(log(atm->p[ip]), atm->z[ip],
41137                 log(atm->p[ip + 1]), atm->z[ip + 1], log(atm->sfp));
41138             zmin = GSL_MAX(zip, zmin);
41139         }
41140     }
41141
41142     /* Check observer altitude... */
41143     if (obs->obsz[ir] < zmin)
41144         ERRMSG("Observer below surface!");
41145
41146     /* Check view point altitude... */
41147     if (obs->vpz[ir] > zmax)
41148         return;
41149
41150     /* Determine Cartesian coordinates for observer and view point... */
41151     geo2cart(obs->obsz[ir], obs->obslon[ir], obs->obslat[ir], xobs);
41152     geo2cart(obs->vpz[ir], obs->vplon[ir], obs->vplat[ir], xvp);
41153
41154     /* Determine initial tangent vector... */
41155     for (int i = 0; i < 3; i++)
41156         ex0[i] = xvp[i] - xobs[i];
41157     norm = NORM(ex0);
41158     for (int i = 0; i < 3; i++)
41159         ex0[i] /= norm;
41160
41161     /* Observer within atmosphere... */
41162     for (int i = 0; i < 3; i++)
41163         x[i] = xobs[i];
41164
41165     /* Observer above atmosphere (search entry point)... */
41166     if (obs->obsz[ir] > zmax) {
41167         double dmax = norm, dmin = 0;
41168         while (fabs(dmin - dmax) > 0.001) {
41169             double d = (dmax + dmin) / 2;
41170             for (int i = 0; i < 3; i++)
41171                 x[i] = xobs[i] + d * ex0[i];
41172             cart2geo(x, &z, &lon, &lat);
41173             if (z <= zmax && z > zmax - 0.001)
41174                 break;
41175             if (z < zmax - 0.0005)
41176                 dmax = d;
41177             else
41178                 dmin = d;
41179         }
41180     }
41181
41182     /* Ray-tracing... */
41183     while (1) {
41184
41185         /* Set step length... */
41186         ds = ctl->rayds;

```

```

04187     if (ctl->raydz > 0) {
04188         norm = NORM(x);
04189         for (int i = 0; i < 3; i++)
04190             xh[i] = x[i] / norm;
04191         double cosa = fabs(DOTP(ex0, xh));
04192         if (cosa != 0)
04193             ds = GSL_MIN(ctl->rayds, ctl->raydz / cosa);
04194     }
04195
04196     /* Determine geolocation... */
04197     cart2geo(x, &z, &lon, &lat);
04198
04199     /* Check if LOS hits the ground or has left atmosphere... */
04200     if (z < zmin || z > zmax) {
04201         stop = (z < zmin ? 2 : 1);
04202         double frac =
04203             ((z <
04204              zmin ? zmin : zmax) - los->z[los->np - 1]) / (z - los->z[los->np -
04205                                                         1]);
04206         geo2cart(los->z[los->np - 1], los->lon[los->np - 1],
04207                 los->lat[los->np - 1], xh);
04208         for (int i = 0; i < 3; i++)
04209             x[i] = xh[i] + frac * (x[i] - xh[i]);
04210         cart2geo(x, &z, &lon, &lat);
04211         los->ds[los->np - 1] = ds * frac;
04212         ds = 0;
04213     }
04214
04215     /* Interpolate atmospheric data... */
04216     intpol_atm(ctl, atm, z, &p, &t, q, k);
04217
04218     /* Save data... */
04219     los->lon[los->np] = lon;
04220     los->lat[los->np] = lat;
04221     los->z[los->np] = z;
04222     los->p[los->np] = p;
04223     los->t[los->np] = t;
04224     for (int ig = 0; ig < ctl->ng; ig++)
04225         los->q[los->np][ig] = q[ig];
04226     for (int id = 0; id < ctl->nd; id++)
04227         los->k[los->np][id] = k[ctl->>window[id]];
04228     los->ds[los->np] = ds;
04229
04230     /* Add cloud extinction... */
04231     if (ctl->ncl > 0 && atm->cldz > 0) {
04232         double aux = exp(-0.5 * POW2((z - atm->clz) / atm->cldz));
04233         for (int id = 0; id < ctl->nd; id++) {
04234             int icl = locate_irr(ctl->clnu, ctl->ncl, ctl->nu[id]);
04235             los->k[los->np][id]
04236                 += aux * LIN(ctl->clnu[icl], atm->clk[icl],
04237                             ctl->clnu[icl + 1], atm->clk[icl + 1], ctl->nu[id]);
04238         }
04239     }
04240
04241     /* Increment and check number of LOS points... */
04242     if ((++los->np) > NLOS)
04243         ERRMSG("Too many LOS points!");
04244
04245     /* Check stop flag... */
04246     if (stop) {
04247
04248         /* Set surface temperature... */
04249         if (ctl->nsf > 0 && atm->sft > 0)
04250             t = atm->sft;
04251         los->sft = (stop == 2 ? t : -999);
04252
04253         /* Set surface emissivity... */
04254         for (int id = 0; id < ctl->nd; id++) {
04255             los->sfeps[id] = 1.0;
04256             if (ctl->nsf > 0) {
04257                 int isf = locate_irr(ctl->sfnu, ctl->nsf, ctl->nu[id]);
04258                 los->sfeps[id] = LIN(ctl->sfnu[isf], atm->sfeps[isf],
04259                                     ctl->sfnu[isf + 1], atm->sfeps[isf + 1],
04260                                     ctl->nu[id]);
04261             }
04262         }
04263
04264         /* Leave raytracer... */
04265         break;
04266     }
04267
04268     /* Determine refractivity... */
04269     if (ctl->refrac && z <= zrefrac)
04270         n = 1 + refractivity(p, t);
04271     else
04272         n = 1;
04273

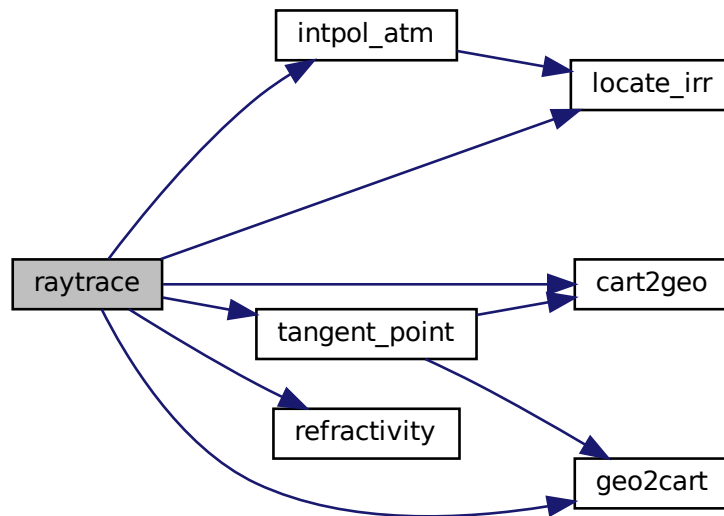
```

```

04274      /* Construct new tangent vector (first term)... */
04275      for (int i = 0; i < 3; i++)
04276          exl[i] = ex0[i] * n;
04277
04278      /* Compute gradient of refractivity... */
04279      if (ctl->refrac && z <= zrefrac) {
04280          for (int i = 0; i < 3; i++)
04281              xh[i] = x[i] + 0.5 * ds * ex0[i];
04282          cart2geo(xh, &z, &lon, &lat);
04283          intpol_atm(ctl, atm, z, &p, &t, q, k);
04284          n = refractivity(p, t);
04285          for (int i = 0; i < 3; i++) {
04286              xh[i] += h;
04287              cart2geo(xh, &z, &lon, &lat);
04288              intpol_atm(ctl, atm, z, &p, &t, q, k);
04289              ng[i] = (refractivity(p, t) - n) / h;
04290              xh[i] -= h;
04291          }
04292      } else
04293          for (int i = 0; i < 3; i++)
04294              ng[i] = 0;
04295
04296      /* Construct new tangent vector (second term)... */
04297      for (int i = 0; i < 3; i++)
04298          exl[i] += ds * ng[i];
04299
04300      /* Normalize new tangent vector... */
04301      norm = NORM(exl);
04302      for (int i = 0; i < 3; i++)
04303          exl[i] /= norm;
04304
04305      /* Determine next point of LOS... */
04306      for (int i = 0; i < 3; i++)
04307          x[i] += 0.5 * ds * (ex0[i] + exl[i]);
04308
04309      /* Copy tangent vector... */
04310      for (int i = 0; i < 3; i++)
04311          ex0[i] = exl[i];
04312  }
04313
04314      /* Get tangent point (to be done before changing segment lengths!)... */
04315      tangent_point(los, &obs->tpz[ir], &obs->tplon[ir], &obs->tplat[ir]);
04316
04317      /* Change segment lengths according to trapezoid rule... */
04318      for (int ip = los->np - 1; ip >= 1; ip--)
04319          los->ds[ip] = 0.5 * (los->ds[ip - 1] + los->ds[ip]);
04320      los->ds[0] *= 0.5;
04321
04322      /* Compute column density... */
04323      for (int ip = 0; ip < los->np; ip++)
04324          for (int ig = 0; ig < ctl->ng; ig++)
04325              los->u[ip][ig] = 10 * los->q[ip][ig] * los->p[ip]
04326                  / (KB * los->t[ip]) * los->ds[ip];
04327  }

```

Here is the call graph for this function:



**5.17.2.35 read\_atm()** void read\_atm (  
     const char \* dirname,  
     const char \* filename,  
     ctl\_t \* ctl,  
     atm\_t \* atm )

Read atmospheric data.

Definition at line 4331 of file [jurassic.c](#).

```

04335     {
04336
04337     FILE *in;
04338
04339     char file[LEN], line[LEN], *tok;
04340
04341     /* Init... */
04342     atm->np = 0;
04343
04344     /* Set filename... */
04345     if (dirname != NULL)
04346         sprintf(file, "%s/%s", dirname, filename);
04347     else
04348         sprintf(file, "%s", filename);
04349
04350     /* Write info... */
04351     LOG(1, "Read atmospheric data: %s", file);
04352
04353     /* Open file... */
04354     if (!(in = fopen(file, "r")))
04355         ERRMSG("Cannot open file!");
04356
04357     /* Read line... */
04358     while (fgets(line, LEN, in)) {
04359
04360         /* Read data... */
04361         TOK(line, tok, "%lg", atm->time[atm->np]);
04362         TOK(NULL, tok, "%lg", atm->z[atm->np]);

```

```

04363     TOK(NULL, tok, "%lg", atm->lon[atm->np]);
04364     TOK(NULL, tok, "%lg", atm->lat[atm->np]);
04365     TOK(NULL, tok, "%lg", atm->p[atm->np]);
04366     TOK(NULL, tok, "%lg", atm->t[atm->np]);
04367     for (int ig = 0; ig < ctl->ng; ig++)
04368         TOK(NULL, tok, "%lg", atm->q[ig][atm->np]);
04369     for (int iw = 0; iw < ctl->nw; iw++)
04370         TOK(NULL, tok, "%lg", atm->k[iw][atm->np]);
04371     if (ctl->ncl > 0 && atm->np == 0) {
04372         TOK(NULL, tok, "%lg", atm->clz);
04373         TOK(NULL, tok, "%lg", atm->cldz);
04374         for (int icl = 0; icl < ctl->ncl; icl++)
04375             TOK(NULL, tok, "%lg", atm->clk[icl]);
04376     }
04377     if (ctl->nsf > 0 && atm->np == 0) {
04378         TOK(NULL, tok, "%lg", atm->sfz);
04379         TOK(NULL, tok, "%lg", atm->sfp);
04380         TOK(NULL, tok, "%lg", atm->sft);
04381         for (int isf = 0; isf < ctl->nsf; isf++)
04382             TOK(NULL, tok, "%lg", atm->sfeps[isf]);
04383     }
04384
04385     /* Increment data point counter... */
04386     if ((++atm->np) > NP)
04387         ERRMSG("Too many data points!");
04388 }
04389
04390 /* Close file... */
04391 fclose(in);
04392
04393 /* Check number of points... */
04394 if (atm->np < 1)
04395     ERRMSG("Could not read any data!");
04396 }

```

**5.17.2.36 read\_ctl()** void read\_ctl (

```

    int argc,
    char * argv[],
    ctl_t * ctl )

```

Read forward model control parameters.

Definition at line 4400 of file [jurassic.c](#).

```

04403     {
04404
04405     /* Write info... */
04406     LOG(1, "\nJuelich Rapid Spectral Simulation Code (JURASSIC)\n"
04407          "(executable: %s | version: %s | compiled: %s, %s)\n",
04408          argv[0], VERSION, __DATE__, __TIME__);
04409
04410     /* Emitters... */
04411     ctl->ng = (int) scan_ctl(argc, argv, "NG", -1, "0", NULL);
04412     if (ctl->ng < 0 || ctl->ng > NG)
04413         ERRMSG("Set 0 <= NG <= MAX!");
04414     for (int ig = 0; ig < ctl->ng; ig++)
04415         scan_ctl(argc, argv, "EMITTER", ig, "", ctl->emitter[ig]);
04416
04417     /* Radiance channels... */
04418     ctl->nd = (int) scan_ctl(argc, argv, "ND", -1, "0", NULL);
04419     if (ctl->nd < 0 || ctl->nd > ND)
04420         ERRMSG("Set 0 <= ND <= MAX!");
04421     for (int id = 0; id < ctl->nd; id++)
04422         ctl->nu[id] = scan_ctl(argc, argv, "NU", id, "", NULL);
04423
04424     /* Spectral windows... */
04425     ctl->nw = (int) scan_ctl(argc, argv, "NW", -1, "1", NULL);
04426     if (ctl->nw < 0 || ctl->nw > NW)
04427         ERRMSG("Set 0 <= NW <= MAX!");
04428     for (int id = 0; id < ctl->nd; id++)
04429         ctl->>window[id] = (int) scan_ctl(argc, argv, "WINDOW", id, "0", NULL);
04430
04431     /* Cloud data... */
04432     ctl->ncl = (int) scan_ctl(argc, argv, "NCL", -1, "0", NULL);
04433     if (ctl->ncl < 0 || ctl->ncl > NCL)
04434         ERRMSG("Set 0 <= NCL <= MAX!");
04435     if (ctl->ncl == 1)
04436         ERRMSG("Set NCL > 1!");
04437     for (int icl = 0; icl < ctl->ncl; icl++)

```

```

04438     ctl->clnu[icl] = scan_ctl(argc, argv, "CLNU", icl, "", NULL);
04439
04440     /* Surface data... */
04441     ctl->nsf = (int) scan_ctl(argc, argv, "NSF", -1, "0", NULL);
04442     if (ctl->nsf < 0 || ctl->nsf > NSF)
04443         ERRMSG("Set 0 <= NSF <= MAX!");
04444     if (ctl->nsf == 1)
04445         ERRMSG("Set NSF > 1!");
04446     for (int isf = 0; isf < ctl->nsf; isf++)
04447         ctl->sfnu[isf] = scan_ctl(argc, argv, "SFNU", isf, "", NULL);
04448     ctl->sftype = (int) scan_ctl(argc, argv, "SFTYPE", -1, "2", NULL);
04449     if (ctl->sftype < 0 || ctl->sftype > 3)
04450         ERRMSG("Set 0 <= SFTYPE <= 3!");
04451     ctl->sfsza = scan_ctl(argc, argv, "SFSZA", -1, "-999", NULL);
04452
04453     /* Emissivity look-up tables... */
04454     scan_ctl(argc, argv, "TBLBASE", -1, "-", ctl->tblbase);
04455     ctl->tblfmt = (int) scan_ctl(argc, argv, "TBLFMT", -1, "1", NULL);
04456
04457     /* Hydrostatic equilibrium... */
04458     ctl->hydZ = scan_ctl(argc, argv, "HYDZ", -1, "-999", NULL);
04459
04460     /* Continua... */
04461     ctl->ctm_co2 = (int) scan_ctl(argc, argv, "CTM_CO2", -1, "1", NULL);
04462     ctl->ctm_h2o = (int) scan_ctl(argc, argv, "CTM_H2O", -1, "1", NULL);
04463     ctl->ctm_n2 = (int) scan_ctl(argc, argv, "CTM_N2", -1, "1", NULL);
04464     ctl->ctm_o2 = (int) scan_ctl(argc, argv, "CTM_O2", -1, "1", NULL);
04465
04466     /* Ray-tracing... */
04467     ctl->refrac = (int) scan_ctl(argc, argv, "REFRAC", -1, "1", NULL);
04468     ctl->rayds = scan_ctl(argc, argv, "RAYDS", -1, "10", NULL);
04469     ctl->raydz = scan_ctl(argc, argv, "RAYDZ", -1, "0.1", NULL);
04470
04471     /* Field of view... */
04472     scan_ctl(argc, argv, "FOV", -1, "-", ctl->fov);
04473
04474     /* Retrieval interface... */
04475     ctl->retp_zmin = scan_ctl(argc, argv, "RETP_ZMIN", -1, "-999", NULL);
04476     ctl->retp_zmax = scan_ctl(argc, argv, "RETP_ZMAX", -1, "-999", NULL);
04477     ctl->rett_zmin = scan_ctl(argc, argv, "RETT_ZMIN", -1, "-999", NULL);
04478     ctl->rett_zmax = scan_ctl(argc, argv, "RETT_ZMAX", -1, "-999", NULL);
04479     for (int ig = 0; ig < ctl->ng; ig++) {
04480         ctl->retq_zmin[ig] = scan_ctl(argc, argv, "RETQ_ZMIN", ig, "-999", NULL);
04481         ctl->retq_zmax[ig] = scan_ctl(argc, argv, "RETQ_ZMAX", ig, "-999", NULL);
04482     }
04483     for (int iw = 0; iw < ctl->nw; iw++) {
04484         ctl->retk_zmin[iw] = scan_ctl(argc, argv, "RETK_ZMIN", iw, "-999", NULL);
04485         ctl->retk_zmax[iw] = scan_ctl(argc, argv, "RETK_ZMAX", iw, "-999", NULL);
04486     }
04487     ctl->ret_clz = (int) scan_ctl(argc, argv, "RET_CLZ", -1, "0", NULL);
04488     ctl->ret_cldz = (int) scan_ctl(argc, argv, "RET_CLDZ", -1, "0", NULL);
04489     ctl->ret_clk = (int) scan_ctl(argc, argv, "RET_CLK", -1, "0", NULL);
04490     ctl->ret_sfz = (int) scan_ctl(argc, argv, "RET_SFZ", -1, "0", NULL);
04491     ctl->ret_sfp = (int) scan_ctl(argc, argv, "RET_SFP", -1, "0", NULL);
04492     ctl->ret_sft = (int) scan_ctl(argc, argv, "RET_SFT", -1, "0", NULL);
04493     ctl->ret_sfeps = (int) scan_ctl(argc, argv, "RET_SFEPS", -1, "0", NULL);
04494
04495     /* Output flags... */
04496     ctl->write_bbt = (int) scan_ctl(argc, argv, "WRITE_BBT", -1, "0", NULL);
04497     ctl->write_matrix =
04498         (int) scan_ctl(argc, argv, "WRITE_MATRIX", -1, "0", NULL);
04499
04500     /* External forward models... */
04501     ctl->formod = (int) scan_ctl(argc, argv, "FORMOD", -1, "1", NULL);
04502     scan_ctl(argc, argv, "RFMBIN", -1, "-", ctl->rfrm-bin);
04503     scan_ctl(argc, argv, "RFMHIT", -1, "-", ctl->rfrm-hit);
04504     for (int ig = 0; ig < ctl->ng; ig++)
04505         scan_ctl(argc, argv, "RFMXSC", ig, "-", ctl->rfrm-xsc[ig]);
04506 }

```

Here is the call graph for this function:



**5.17.2.37 read\_matrix()** void read\_matrix (  
     const char \* *dirname*,  
     const char \* *filename*,  
     gsl\_matrix \* *matrix* )

Read matrix.

Definition at line 4510 of file [jurassic.c](#).

```
04513     {
04514
04515     FILE *in;
04516
04517     char dum[LEN], file[LEN], line[LEN];
04518
04519     double value;
04520
04521     int i, j;
04522
04523     /* Set filename... */
04524     if (dirname != NULL)
04525         sprintf(file, "%s/%s", dirname, filename);
04526     else
04527         sprintf(file, "%s", filename);
04528
04529     /* Write info... */
04530     LOG(1, "Read matrix: %s", file);
04531
04532     /* Open file... */
04533     if (!(in = fopen(file, "r")))
04534         ERRMSG("Cannot open file!");
04535
04536     /* Read data... */
04537     gsl_matrix_set_zero(matrix);
04538     while (fgets(line, LEN, in))
04539         if (sscanf(line, "%d %s %s %s %s %s %d %s %s %s %s %s %lg",
04540                 &i, dum, dum, dum, dum, dum,
04541                 &j, dum, dum, dum, dum, dum, &value) == 13)
04542             gsl_matrix_set(matrix, (size_t) i, (size_t) j, value);
04543
04544     /* Close file... */
04545     fclose(in);
04546 }
```

**5.17.2.38 read\_obs()** void read\_obs (  
     const char \* *dirname*,  
     const char \* *filename*,  
     ctl\_t \* *ctl*,  
     obs\_t \* *obs* )

Read observation data.

Definition at line 4550 of file [jurassic.c](#).

```
04554     {
04555
04556     FILE *in;
04557
04558     char file[LEN], line[LEN], *tok;
04559
04560     /* Init... */
04561     obs->nr = 0;
04562
04563     /* Set filename... */
04564     if (dirname != NULL)
04565         sprintf(file, "%s/%s", dirname, filename);
04566     else
04567         sprintf(file, "%s", filename);
04568
04569     /* Write info... */
04570     LOG(1, "Read observation data: %s", file);
```

```

04571
04572  /* Open file... */
04573  if (!(in = fopen(file, "r")))
04574      ERRMSG("Cannot open file!");
04575
04576  /* Read line... */
04577  while (fgets(line, LEN, in)) {
04578
04579      /* Read data... */
04580      TOK(line, tok, "%lg", obs->time[obs->nr]);
04581      TOK(NULL, tok, "%lg", obs->obsz[obs->nr]);
04582      TOK(NULL, tok, "%lg", obs->obslon[obs->nr]);
04583      TOK(NULL, tok, "%lg", obs->obslat[obs->nr]);
04584      TOK(NULL, tok, "%lg", obs->vpz[obs->nr]);
04585      TOK(NULL, tok, "%lg", obs->vplon[obs->nr]);
04586      TOK(NULL, tok, "%lg", obs->vplat[obs->nr]);
04587      TOK(NULL, tok, "%lg", obs->tpz[obs->nr]);
04588      TOK(NULL, tok, "%lg", obs->tplon[obs->nr]);
04589      TOK(NULL, tok, "%lg", obs->tplat[obs->nr]);
04590      for (int id = 0; id < ctl->nd; id++)
04591          TOK(NULL, tok, "%lg", obs->rad[id][obs->nr]);
04592      for (int id = 0; id < ctl->nd; id++)
04593          TOK(NULL, tok, "%lg", obs->tau[id][obs->nr]);
04594
04595      /* Increment counter... */
04596      if ((++obs->nr) > NR)
04597          ERRMSG("Too many rays!");
04598  }
04599
04600  /* Close file... */
04601  fclose(in);
04602
04603  /* Check number of points... */
04604  if (obs->nr < 1)
04605      ERRMSG("Could not read any data!");
04606 }

```

**5.17.2.39 read\_obs\_rfm()** double read\_obs\_rfm (

```

    const char * basename,
    double z,
    double * nu,
    double * f,
    int n )

```

Read observation data in RFM format.

Definition at line 4610 of file [jurassic.c](#).

```

04615  {
04616
04617  FILE *in;
04618
04619  char filename[LEN];
04620
04621  double filt, fsum = 0, nu2[NSHAPE], *nurfm, *rad, radsum = 0;
04622
04623  int i, idx, ipt, npts;
04624
04625  /* Allocate... */
04626  ALLOC(nurfm, double,
04627        RFMNPTS);
04628  ALLOC(rad, double,
04629        RFMNPTS);
04630
04631  /* Search RFM spectrum... */
04632  sprintf(filename, "%s_%05d.asc", basename, (int) (z * 1000));
04633  if (!(in = fopen(filename, "r"))) {
04634      sprintf(filename, "%s_%05d.asc", basename, (int) (z * 1000) + 1);
04635      if (!(in = fopen(filename, "r")))
04636          ERRMSG("Cannot find RFM data file!");
04637  }
04638  fclose(in);
04639
04640  /* Read RFM spectrum... */
04641  read_rfm_spec(filename, nurfm, rad, &npts);
04642
04643  /* Set wavenumbers... */
04644  nu2[0] = nu[0];

```

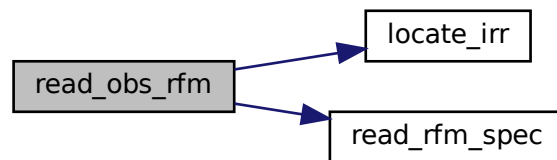


```

04645     nu2[n - 1] = nu[n - 1];
04646     for (i = 1; i < n - 1; i++)
04647         nu2[i] = LIN(0.0, nu2[0], n - 1.0, nu2[n - 1], i);
04648
04649     /* Convolute... */
04650     for (ipts = 0; ipts < npts; ipts++)
04651         if (nurfm[ipts] >= nu2[0] && nurfm[ipts] <= nu2[n - 1]) {
04652             idx = locate_irr(nu2, n, nurfm[ipts]);
04653             filt = LIN(nu2[idx], f[idx], nu2[idx + 1], f[idx + 1], nurfm[ipts]);
04654             fsum += filt;
04655             radsum += filt * rad[ipts];
04656         }
04657
04658     /* Free... */
04659     free(nurfm);
04660     free(rad);
04661
04662     /* Return radiance... */
04663     return radsum / fsum;
04664 }

```

Here is the call graph for this function:



**5.17.2.40 read\_rfm\_spec()** void read\_rfm\_spec (

```

    const char * filename,
    double * nu,
    double * rad,
    int * npts )

```

Read RFM spectrum.

Definition at line 4668 of file [jurassic.c](#).

```

04672     {
04673
04674         FILE *in;
04675
04676         char line[RFMLINE], *tok;
04677
04678         double dnu, nu0, nul;
04679
04680         int i, ipts = 0;
04681
04682         /* Write info... */
04683         printf("Read RFM data: %s\n", filename);
04684
04685         /* Open file... */
04686         if (!(in = fopen(filename, "r")))
04687             ERRMSG("Cannot open file!");
04688
04689         /* Read header..... */
04690         for (i = 0; i < 4; i++)
04691             if (fgets(line, RFMLINE, in) == NULL)
04692                 ERRMSG("Error while reading file header!");
04693         sscanf(line, "%d %lg %lg %lg", npts, &nu0, &dnu, &nul);
04694         if (*npts > RFMNPTS)

```

```

04695     ERRMSG("Too many spectral grid points!");
04696
04697     /* Read radiance data... */
04698     while (fgets(line, RFMLINE, in) && ipts < *npts - 1) {
04699         if ((tok = strtok(line, " \t\n")) != NULL)
04700             if (sscanf(tok, "%lg", &rad[ipts]) == 1)
04701                 ipts++;
04702         while ((tok = strtok(NULL, " \t\n")) != NULL)
04703             if (sscanf(tok, "%lg", &rad[ipts]) == 1)
04704                 ipts++;
04705     }
04706     if (ipts != *npts)
04707         ERRMSG("Error while reading RFM data!");
04708
04709     /* Compute wavenumbers... */
04710     for (ipts = 0; ipts < *npts; ipts++)
04711         nu[ipts] = LIN(0.0, nu0, (double) (*npts - 1), nul, (double) ipts);
04712
04713     /* Close file... */
04714     fclose(in);
04715 }

```

#### 5.17.2.41 read\_shape() void read\_shape (

```

    const char * filename,
    double * x,
    double * y,
    int * n )

```

Read shape function.

Definition at line 4719 of file [jurassic.c](#).

```

04723     {
04724
04725     FILE *in;
04726
04727     char line[LEN];
04728
04729     /* Write info... */
04730     LOG(1, "Read shape function: %s", filename);
04731
04732     /* Open file... */
04733     if (!(in = fopen(filename, "r")))
04734         ERRMSG("Cannot open file!");
04735
04736     /* Read data... */
04737     *n = 0;
04738     while (fgets(line, LEN, in))
04739         if (sscanf(line, "%lg %lg", &x[*n], &y[*n]) == 2)
04740             if (++(*n) > NSHAPE)
04741                 ERRMSG("Too many data points!");
04742
04743     /* Check number of points... */
04744     if (*n < 1)
04745         ERRMSG("Could not read any data!");
04746
04747     /* Close file... */
04748     fclose(in);
04749 }

```

#### 5.17.2.42 read\_tbl() void read\_tbl (

```

    ctl_t * ctl,
    tbl_t * tbl )

```

Read look-up table data.

Definition at line 4753 of file [jurassic.c](#).

```

04755     {
04756
04757     FILE *in;

```

```

04758
04759 char filename[2 * LEN], line[LEN];
04760
04761 double eps, press, temp, u;
04762
04763 /* Loop over trace gases and channels... */
04764 for (int id = 0; id < ctl->nd; id++)
04765     for (int ig = 0; ig < ctl->ng; ig++) {
04766
04767         /* Initialize... */
04768         tbl->np[id][ig] = -1;
04769         double eps_old = -999;
04770         double press_old = -999;
04771         double temp_old = -999;
04772         double u_old = -999;
04773
04774         /* Set filename... */
04775         sprintf(filename, "%s_%.4f_%.s", ctl->tblbase,
04776             ctl->nu[id], ctl->emitter[ig],
04777             ctl->tblfmt == 1 ? "tab" : "bin");
04778
04779         /* Write info... */
04780         LOG(1, "Read emissivity table: %s", filename);
04781
04782         /* Try to open file... */
04783         if (!(in = fopen(filename, "r"))) {
04784             WARN("Missing emissivity table: %s", filename);
04785             continue;
04786         }
04787
04788         /* Read ASCII tables... */
04789         if (ctl->tblfmt == 1) {
04790
04791             /* Read data... */
04792             while (fgets(line, LEN, in)) {
04793
04794                 /* Parse line... */
04795                 if (sscanf(line, "%lg %lg %lg %lg", &press, &temp, &u, &eps) != 4)
04796                     continue;
04797
04798                 /* Check ranges... */
04799                 if (u < 0 || u > 1e30 || eps < 0 || eps > 1)
04800                     continue;
04801
04802                 /* Determine pressure index... */
04803                 if (press != press_old) {
04804                     press_old = press;
04805                     if (++tbl->np[id][ig] >= TBLNP)
04806                         ERRMSG("Too many pressure levels!");
04807                     tbl->nt[id][ig][tbl->np[id][ig]] = -1;
04808                 }
04809
04810                 /* Determine temperature index... */
04811                 if (temp != temp_old) {
04812                     temp_old = temp;
04813                     if (++tbl->nt[id][ig][tbl->np[id][ig]] >= TBLNT)
04814                         ERRMSG("Too many temperatures!");
04815                     tbl->nu[id][ig][tbl->np[id][ig]]
04816                         [tbl->nt[id][ig][tbl->np[id][ig]]] = -1;
04817                 }
04818
04819                 /* Determine column density index... */
04820                 if ((eps > eps_old && u > u_old) || tbl->nu[id][ig][tbl->np[id][ig]]
04821                     [tbl->nt[id][ig][tbl->np[id][ig]]] < 0) {
04822                     eps_old = eps;
04823                     u_old = u;
04824                     if (++tbl->nu[id][ig][tbl->np[id][ig]]
04825                         [tbl->nt[id][ig][tbl->np[id][ig]]] >= TBLNU) {
04826                         tbl->nu[id][ig][tbl->np[id][ig]]
04827                             [tbl->nt[id][ig][tbl->np[id][ig]]]--;
04828                         continue;
04829                     }
04830                 }
04831
04832                 /* Store data... */
04833                 tbl->p[id][ig][tbl->np[id][ig]] = press;
04834                 tbl->t[id][ig][tbl->np[id][ig]][tbl->nt[id][ig][tbl->np[id][ig]]]
04835                     = temp;
04836                 tbl->u[id][ig][tbl->np[id][ig]][tbl->nt[id][ig][tbl->np[id][ig]]]
04837                     [tbl->nu[id][ig][tbl->np[id][ig]]]
04838                     [tbl->nt[id][ig][tbl->np[id][ig]]] = (float) u;
04839                 tbl->eps[id][ig][tbl->np[id][ig]][tbl->nt[id][ig][tbl->np[id][ig]]]
04840                     [tbl->nu[id][ig][tbl->np[id][ig]]]
04841                     [tbl->nt[id][ig][tbl->np[id][ig]]] = (float) eps;
04842             }
04843
04844         /* Increment counters... */

```

```

04845     tbl->np[id][ig]++;
04846     for (int ip = 0; ip < tbl->np[id][ig]; ip++) {
04847         tbl->nt[id][ig][ip]++;
04848         for (int it = 0; it < tbl->nt[id][ig][ip]; it++)
04849             tbl->nu[id][ig][ip][it]++;
04850     }
04851 }
04852
04853 /* Read binary data... */
04854 else if (ctl->tblfmt == 2) {
04855
04856     /* Read data... */
04857     FREAD(&tbl->np[id][ig], int,
04858          1,
04859          in);
04860     if (tbl->np[id][ig] > TBLNP)
04861         ERRMSG("Too many pressure levels!");
04862     FREAD(tbl->p[id][ig], double,
04863          (size_t) tbl->np[id][ig],
04864          in);
04865     for (int ip = 0; ip < tbl->np[id][ig]; ip++) {
04866         FREAD(&tbl->nt[id][ig][ip], int,
04867              1,
04868              in);
04869         if (tbl->nt[id][ig][ip] > TBLNT)
04870             ERRMSG("Too many temperatures!");
04871         FREAD(tbl->t[id][ig][ip], double,
04872              (size_t) tbl->nt[id][ig][ip],
04873              in);
04874         for (int it = 0; it < tbl->nt[id][ig][ip]; it++) {
04875             FREAD(&tbl->nu[id][ig][ip][it], int,
04876                  1,
04877                  in);
04878             if (tbl->nu[id][ig][ip][it] > TBLNU)
04879                 ERRMSG("Too many column densities!");
04880             FREAD(tbl->u[id][ig][ip][it], float,
04881                  (size_t) tbl->nu[id][ig][ip][it],
04882                  in);
04883             FREAD(tbl->eps[id][ig][ip][it], float,
04884                  (size_t) tbl->nu[id][ig][ip][it],
04885                  in);
04886         }
04887     }
04888 }
04889
04890 /* Error message... */
04891 else
04892     ERRMSG("Unknown look-up table format!");
04893
04894 /* Close file... */
04895 fclose(in);
04896 }
04897 }

```

**5.17.2.43 refractivity()** double refractivity (  
double p,  
double t )

Compute refractivity (return value is n - 1).

Definition at line 4901 of file [jurassic.c](#).

```

04903     {
04904
04905     /* Refractivity of air at 4 to 15 micron... */
04906     return 7.753e-05 * p / t;
04907 }

```

**5.17.2.44 scan\_ctl()** double scan\_ctl (  
int argc,  
char \* argv[],  
const char \* varname,

```

    int arridx,
    const char * defvalue,
    char * value )

```

Search control parameter file for variable entry.

Definition at line 4911 of file [jurassic.c](#).

```

04917     {
04918
04919     FILE *in = NULL;
04920
04921     char dummy[LEN], fullname1[LEN], fullname2[LEN], line[LEN],
04922         rvarname[LEN], rval[LEN];
04923
04924     int contain = 0;
04925
04926     /* Open file... */
04927     if (argv[1][0] != '-')
04928         if (!(in = fopen(argv[1], "r")))
04929             ERRMSG("Cannot open file!");
04930
04931     /* Set full variable name... */
04932     if (arridx >= 0) {
04933         sprintf(fullname1, "%s[%d]", varname, arridx);
04934         sprintf(fullname2, "%s[*]", varname);
04935     } else {
04936         sprintf(fullname1, "%s", varname);
04937         sprintf(fullname2, "%s", varname);
04938     }
04939
04940     /* Read data... */
04941     if (in != NULL)
04942         while (fgets(line, LEN, in))
04943             if (sscanf(line, "%s %s %s", rvarname, dummy, rval) == 3)
04944                 if (strcasecmp(rvarname, fullname1) == 0 ||
04945                     strcasecmp(rvarname, fullname2) == 0) {
04946                     contain = 1;
04947                     break;
04948                 }
04949     for (int i = 1; i < argc - 1; i++)
04950         if (strcasecmp(argv[i], fullname1) == 0 ||
04951             strcasecmp(argv[i], fullname2) == 0) {
04952             sprintf(rval, "%s", argv[i + 1]);
04953             contain = 1;
04954             break;
04955         }
04956
04957     /* Close file... */
04958     if (in != NULL)
04959         fclose(in);
04960
04961     /* Check for missing variables... */
04962     if (!contain) {
04963         if (strlen(defvalue) > 0)
04964             sprintf(rval, "%s", defvalue);
04965         else
04966             ERRMSG("Missing variable %s!\n", fullname1);
04967     }
04968
04969     /* Write info... */
04970     LOG(1, "%s = %s", fullname1, rval);
04971
04972     /* Return values... */
04973     if (value != NULL)
04974         sprintf(value, "%s", rval);
04975     return atof(rval);
04976 }

```

**5.17.2.45** **sza()** double sza (  
     double sec,  
     double lon,  
     double lat )

Calculate solar zenith angle.

Definition at line 4980 of file [jurassic.c](#).

```

04983         {
04984
04985         /* Number of days and fraction with respect to 2000-01-01T12:00Z... */
04986         double D = sec / 86400 - 0.5;
04987
04988         /* Geocentric apparent ecliptic longitude [rad]... */
04989         double g = (357.529 + 0.98560028 * D) * M_PI / 180;
04990         double q = 280.459 + 0.98564736 * D;
04991         double L = (q + 1.915 * sin(g) + 0.020 * sin(2 * g)) * M_PI / 180;
04992
04993         /* Mean obliquity of the ecliptic [rad]... */
04994         double e = (23.439 - 0.00000036 * D) * M_PI / 180;
04995
04996         /* Declination [rad]... */
04997         double dec = asin(sin(e) * sin(L));
04998
04999         /* Right ascension [rad]... */
05000         double ra = atan2(cos(e) * sin(L), cos(L));
05001
05002         /* Greenwich Mean Sidereal Time [h]... */
05003         double GMST = 18.697374558 + 24.06570982441908 * D;
05004
05005         /* Local Sidereal Time [h]... */
05006         double LST = GMST + lon / 15;
05007
05008         /* Hour angle [rad]... */
05009         double h = LST / 12 * M_PI - ra;
05010
05011         /* Convert latitude... */
05012         lat *= M_PI / 180;
05013
05014         /* Return solar zenith angle [deg]... */
05015         return acos(sin(lat) * sin(dec) +
05016                   cos(lat) * cos(dec) * cos(h)) * 180 / M_PI;
05017 }

```

**5.17.2.46 tangent\_point()** void tangent\_point (

```

    los_t * los,
    double * tpz,
    double * tplon,
    double * tplat )

```

Find tangent point of a given LOS.

Definition at line 5021 of file [jurassic.c](#).

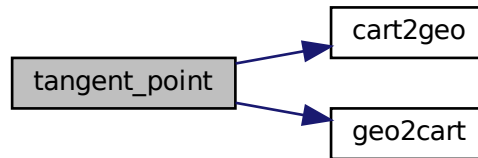
```

05025         {
05026
05027         double dummy, v[3], v0[3], v2[3];
05028
05029         /* Find minimum altitude... */
05030         size_t ip = gsl_stats_min_index(los->z, 1, (size_t) los->np);
05031
05032         /* Nadir or zenith... */
05033         if (ip <= 0 || ip >= (size_t) los->np - 1) {
05034             *tpz = los->z[los->np - 1];
05035             *tplon = los->lon[los->np - 1];
05036             *tplat = los->lat[los->np - 1];
05037         }
05038
05039         /* Limb... */
05040         else {
05041
05042             /* Determine interpolating polynomial y=a*x^2+b*x+c... */
05043             double yy0 = los->z[ip - 1];
05044             double yy1 = los->z[ip];
05045             double yy2 = los->z[ip + 1];
05046             double x1 = sqrt(POW2(los->ds[ip]) - POW2(yy1 - yy0));
05047             double x2 = x1 + sqrt(POW2(los->ds[ip + 1]) - POW2(yy2 - yy1));
05048             double a = 1 / (x1 - x2) * (-(yy0 - yy1) / x1 + (yy0 - yy2) / x2);
05049             double b = -(yy0 - yy1) / x1 - a * x1;
05050             double c = yy0;
05051
05052             /* Get tangent point location... */
05053             double x = -b / (2 * a);
05054             *tpz = a * x * x + b * x + c;
05055             geo2cart(los->z[ip - 1], los->lon[ip - 1], los->lat[ip - 1], v0);
05056             geo2cart(los->z[ip + 1], los->lon[ip + 1], los->lat[ip + 1], v2);

```

```
05057     for (int i = 0; i < 3; i++)
05058         v[i] = LIN(0.0, v0[i], x2, v2[i], x);
05059     cart2geo(v, &dummy, tplon, tplat);
05060 }
05061 }
```

Here is the call graph for this function:



```
5.17.2.47 time2jsec() void time2jsec (
    int year,
    int mon,
    int day,
    int hour,
    int min,
    int sec,
    double remain,
    double * jsec )
```

Convert date to seconds.

Definition at line 5065 of file [jurassic.c](#).

```
05073     {
05074
05075     struct tm t0, t1;
05076
05077     t0.tm_year = 100;
05078     t0.tm_mon = 0;
05079     t0.tm_mday = 1;
05080     t0.tm_hour = 0;
05081     t0.tm_min = 0;
05082     t0.tm_sec = 0;
05083
05084     t1.tm_year = year - 1900;
05085     t1.tm_mon = mon - 1;
05086     t1.tm_mday = day;
05087     t1.tm_hour = hour;
05088     t1.tm_min = min;
05089     t1.tm_sec = sec;
05090
05091     *jsec = (double) timegm(&t1) - (double) timegm(&t0) + remain;
05092 }
```

**5.17.2.48 timer()** void timer (  
     const char \* name,  
     const char \* file,  
     const char \* func,  
     int line,  
     int mode )

Measure wall-clock time.

Definition at line 5096 of file [jurassic.c](#).

```
05101     {
05102
05103     static double w0[10];
05104
05105     static int l0[10], nt;
05106
05107     /* Start new timer... */
05108     if (mode == 1) {
05109         w0[nt] = omp_get_wtime();
05110         l0[nt] = line;
05111         if ((++nt) >= 10)
05112             ERRMSG("Too many timers!");
05113     }
05114
05115     /* Write elapsed time... */
05116     else {
05117
05118         /* Check timer index... */
05119         if (nt - 1 < 0)
05120             ERRMSG("Coding error!");
05121
05122         /* Write elapsed time... */
05123         LOG(1, "Timer '%s' (%s, %s, l%d-%d): %.3f sec",
05124             name, file, func, l0[nt - 1], line, omp_get_wtime() - w0[nt - 1]);
05125     }
05126
05127     /* Stop timer... */
05128     if (mode == 3)
05129         nt--;
05130 }
```

**5.17.2.49 write\_atm()** void write\_atm (  
     const char \* dirname,  
     const char \* filename,  
     ctl\_t \* ctl,  
     atm\_t \* atm )

Write atmospheric data.

Definition at line 5134 of file [jurassic.c](#).

```
05138     {
05139
05140     FILE *out;
05141
05142     char file[LEN];
05143
05144     int n = 6;
05145
05146     /* Set filename... */
05147     if (dirname != NULL)
05148         sprintf(file, "%s/%s", dirname, filename);
05149     else
05150         sprintf(file, "%s", filename);
05151
05152     /* Write info... */
05153     LOG(1, "Write atmospheric data: %s", file);
05154
05155     /* Create file... */
05156     if (!(out = fopen(file, "w")))
05157         ERRMSG("Cannot create file!");
05158
05159     /* Write header... */
05160     fprintf(out,
```



```

05161         "# $1 = time (seconds since 2000-01-01T00:00Z)\n"
05162         "# $2 = altitude [km]\n"
05163         "# $3 = longitude [deg]\n"
05164         "# $4 = latitude [deg]\n"
05165         "# $5 = pressure [hPa]\n" "# $6 = temperature [K]\n");
05166 for (int ig = 0; ig < ctl->ng; ig++)
05167     fprintf(out, "# $d = %s volume mixing ratio [ppv]\n",
05168             ++n, ctl->emitter[ig]);
05169 for (int iw = 0; iw < ctl->nw; iw++)
05170     fprintf(out, "# $d = extinction (window %d) [1/km]\n", ++n, iw);
05171 if (ctl->ncl > 0) {
05172     fprintf(out, "# $d = cloud layer height [km]\n", ++n);
05173     fprintf(out, "# $d = cloud layer depth [km]\n", ++n);
05174     for (int icl = 0; icl < ctl->ncl; icl++)
05175         fprintf(out, "# $d = cloud layer extinction (%.4f cm^-1) [1/km]\n",
05176                 ++n, ctl->clnu[icl]);
05177 }
05178 if (ctl->nsf > 0) {
05179     fprintf(out, "# $d = surface layer height [km]\n", ++n);
05180     fprintf(out, "# $d = surface layer pressure [hPa]\n", ++n);
05181     fprintf(out, "# $d = surface layer temperature [K]\n", ++n);
05182     for (int isf = 0; isf < ctl->nsf; isf++)
05183         fprintf(out, "# $d = surface layer emissivity (%.4f cm^-1)\n",
05184                 ++n, ctl->sfnu[isf]);
05185 }
05186
05187 /* Write data... */
05188 for (int ip = 0; ip < atm->np; ip++) {
05189     if (ip == 0 || atm->time[ip] != atm->time[ip - 1])
05190         fprintf(out, "\n");
05191     fprintf(out, "%.2f %g %g %g %g %g", atm->time[ip], atm->z[ip],
05192             atm->lon[ip], atm->lat[ip], atm->p[ip], atm->t[ip]);
05193     for (int ig = 0; ig < ctl->ng; ig++)
05194         fprintf(out, " %g", atm->q[ig][ip]);
05195     for (int iw = 0; iw < ctl->nw; iw++)
05196         fprintf(out, " %g", atm->k[iw][ip]);
05197     if (ctl->ncl > 0) {
05198         fprintf(out, " %g %g", atm->clz, atm->cldz);
05199         for (int icl = 0; icl < ctl->ncl; icl++)
05200             fprintf(out, " %g", atm->clk[icl]);
05201     }
05202     if (ctl->nsf > 0) {
05203         fprintf(out, " %g %g %g", atm->sfz, atm->sfp, atm->sft);
05204         for (int isf = 0; isf < ctl->nsf; isf++)
05205             fprintf(out, " %g", atm->sfeps[isf]);
05206     }
05207     fprintf(out, "\n");
05208 }
05209
05210 /* Close file... */
05211 fclose(out);
05212 }

```

**5.17.2.50 write\_atm\_rfm()** void write\_atm\_rfm (  
     const char \* filename,  
     ctl\_t \* ctl,  
     atm\_t \* atm )

Write atmospheric data in RFM format.

Definition at line 5216 of file [jurassic.c](#).

```

05219     {
05220
05221     FILE *out;
05222
05223     int ig, ip;
05224
05225     /* Write info... */
05226     printf("Write RFM data: %s\n", filename);
05227
05228     /* Create file... */
05229     if (!(out = fopen(filename, "w")))
05230         ERRMSG("Cannot create file!");
05231
05232     /* Write data... */
05233     fprintf(out, "%d\n", atm->np);
05234     fprintf(out, "%HGT [km]\n");
05235     for (ip = 0; ip < atm->np; ip++)

```

```

05236     fprintf(out, "%g\n", atm->z[ip]);
05237     fprintf(out, "*PRE [mb]\n");
05238     for (ip = 0; ip < atm->np; ip++)
05239         fprintf(out, "%g\n", atm->p[ip]);
05240     fprintf(out, "*TEM [K]\n");
05241     for (ip = 0; ip < atm->np; ip++)
05242         fprintf(out, "%g\n", atm->t[ip]);
05243     for (ig = 0; ig < ctl->ng; ig++) {
05244         fprintf(out, "%s [ppmv]\n", ctl->emitter[ig]);
05245         for (ip = 0; ip < atm->np; ip++)
05246             fprintf(out, "%g\n", atm->q[ig][ip] * 1e6);
05247     }
05248     fprintf(out, "END\n");
05249
05250     /* Close file... */
05251     fclose(out);
05252 }

```

#### 5.17.2.51 write\_matrix() void write\_matrix (

```

    const char * dirname,
    const char * filename,
    ctl_t * ctl,
    gsl_matrix * matrix,
    atm_t * atm,
    obs_t * obs,
    const char * rowspace,
    const char * colspace,
    const char * sort )

```

Write matrix.

Definition at line 5256 of file [jurassic.c](#).

```

05265     {
05266
05267     FILE *out;
05268
05269     char file[LEN], quantity[LEN];
05270
05271     int *cida, *ciqa, *cipa, *cira, *rida, *riqa, *ripa, *rira;
05272
05273     size_t i, j, nc, nr;
05274
05275     /* Check output flag... */
05276     if (!ctl->write_matrix)
05277         return;
05278
05279     /* Allocate... */
05280     ALLOC(cida, int,
05281           M);
05282     ALLOC(ciqa, int,
05283           N);
05284     ALLOC(cipa, int,
05285           N);
05286     ALLOC(cira, int,
05287           M);
05288     ALLOC(rida, int,
05289           M);
05290     ALLOC(riqa, int,
05291           N);
05292     ALLOC(ripa, int,
05293           N);
05294     ALLOC(rira, int,
05295           M);
05296
05297     /* Set filename... */
05298     if (dirname != NULL)
05299         sprintf(file, "%s/%s", dirname, filename);
05300     else
05301         sprintf(file, "%s", filename);
05302
05303     /* Write info... */
05304     LOG(1, "Write matrix: %s", file);
05305
05306     /* Create file... */
05307     if (!(out = fopen(file, "w")))

```

```

05308     ERRMSG("Cannot create file!");
05309
05310     /* Write header (row space)... */
05311     if (row_space[0] == 'y') {
05312
05313         fprintf(out,
05314             "# $1 = Row: index (measurement space)\n"
05315             "# $2 = Row: channel wavenumber [cm^-1]\n"
05316             "# $3 = Row: time (seconds since 2000-01-01T00:00Z)\n"
05317             "# $4 = Row: view point altitude [km]\n"
05318             "# $5 = Row: view point longitude [deg]\n"
05319             "# $6 = Row: view point latitude [deg]\n");
05320
05321         /* Get number of rows... */
05322         nr = obs2y(ctl, obs, NULL, rida, rira);
05323
05324     } else {
05325
05326         fprintf(out,
05327             "# $1 = Row: index (state space)\n"
05328             "# $2 = Row: name of quantity\n"
05329             "# $3 = Row: time (seconds since 2000-01-01T00:00Z)\n"
05330             "# $4 = Row: altitude [km]\n"
05331             "# $5 = Row: longitude [deg]\n" "# $6 = Row: latitude [deg]\n");
05332
05333         /* Get number of rows... */
05334         nr = atm2x(ctl, atm, NULL, rira, ripa);
05335     }
05336
05337     /* Write header (column space)... */
05338     if (col_space[0] == 'y') {
05339
05340         fprintf(out,
05341             "# $7 = Col: index (measurement space)\n"
05342             "# $8 = Col: channel wavenumber [cm^-1]\n"
05343             "# $9 = Col: time (seconds since 2000-01-01T00:00Z)\n"
05344             "# $10 = Col: view point altitude [km]\n"
05345             "# $11 = Col: view point longitude [deg]\n"
05346             "# $12 = Col: view point latitude [deg]\n");
05347
05348         /* Get number of columns... */
05349         nc = obs2y(ctl, obs, NULL, cida, cira);
05350
05351     } else {
05352
05353         fprintf(out,
05354             "# $7 = Col: index (state space)\n"
05355             "# $8 = Col: name of quantity\n"
05356             "# $9 = Col: time (seconds since 2000-01-01T00:00Z)\n"
05357             "# $10 = Col: altitude [km]\n"
05358             "# $11 = Col: longitude [deg]\n" "# $12 = Col: latitude [deg]\n");
05359
05360         /* Get number of columns... */
05361         nc = atm2x(ctl, atm, NULL, cira, cipa);
05362     }
05363
05364     /* Write header entry... */
05365     fprintf(out, "# $13 = Matrix element\n\n");
05366
05367     /* Write matrix data... */
05368     i = j = 0;
05369     while (i < nr && j < nc) {
05370
05371         /* Write info about the row... */
05372         if (row_space[0] == 'y')
05373             fprintf(out, "%d %.4f %.2f %g %g %g",
05374                 (int) i, ctl->nu[rida[i]],
05375                 obs->time[rira[i]], obs->vpz[rira[i]],
05376                 obs->vplon[rira[i]], obs->vplat[rira[i]]);
05377         else {
05378             idx2name(ctl, rira[i], quantity);
05379             fprintf(out, "%d %s %.2f %g %g %g", (int) i, quantity,
05380                 atm->time[ripa[i]], atm->z[ripa[i]],
05381                 atm->lon[ripa[i]], atm->lat[ripa[i]]);
05382         }
05383
05384         /* Write info about the column... */
05385         if (col_space[0] == 'y')
05386             fprintf(out, " %d %.4f %.2f %g %g %g",
05387                 (int) j, ctl->nu[cida[j]],
05388                 obs->time[cira[j]], obs->vpz[cira[j]],
05389                 obs->vplon[cira[j]], obs->vplat[cira[j]]);
05390         else {
05391             idx2name(ctl, cira[j], quantity);
05392             fprintf(out, " %d %s %.2f %g %g %g", (int) j, quantity,
05393                 atm->time[cipa[j]], atm->z[cipa[j]],
05394                 atm->lon[cipa[j]], atm->lat[cipa[j]]);

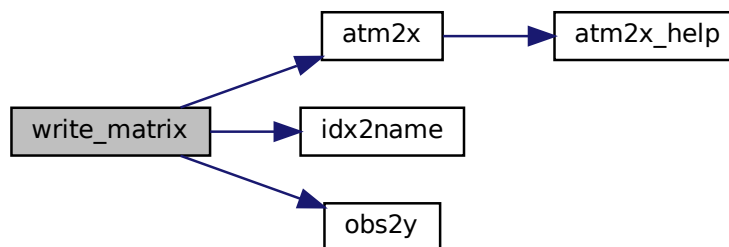
```

```

05395     }
05396
05397     /* Write matrix entry... */
05398     fprintf(out, " %g\n", gsl_matrix_get(matrix, i, j));
05399
05400     /* Set matrix indices... */
05401     if (sort[0] == 'r') {
05402         j++;
05403         if (j >= nc) {
05404             j = 0;
05405             i++;
05406             fprintf(out, "\n");
05407         }
05408     } else {
05409         i++;
05410         if (i >= nr) {
05411             i = 0;
05412             j++;
05413             fprintf(out, "\n");
05414         }
05415     }
05416 }
05417
05418 /* Close file... */
05419 fclose(out);
05420
05421 /* Free... */
05422 free(cida);
05423 free(ciga);
05424 free(cipa);
05425 free(cira);
05426 free(rida);
05427 free(riqa);
05428 free(ripa);
05429 free(rira);
05430 }

```

Here is the call graph for this function:



**5.17.2.52 write\_obs()** void write\_obs (

```

    const char * dirname,
    const char * filename,
    ctl_t * ctl,
    obs_t * obs )

```

Write observation data.

Definition at line 5434 of file [jurassic.c](#).

```

05438     {
05439
05440     FILE *out;

```

```

05441
05442     char file[LEN];
05443
05444     int n = 10;
05445
05446     /* Set filename... */
05447     if (dirname != NULL)
05448         sprintf(file, "%s/%s", dirname, filename);
05449     else
05450         sprintf(file, "%s", filename);
05451
05452     /* Write info... */
05453     LOG(1, "Write observation data: %s", file);
05454
05455     /* Create file... */
05456     if (!(out = fopen(file, "w")))
05457         ERRMSG("Cannot create file!");
05458
05459     /* Write header... */
05460     fprintf(out,
05461             "# $1 = time (seconds since 2000-01-01T00:00Z)\n"
05462             "# $2 = observer altitude [km]\n"
05463             "# $3 = observer longitude [deg]\n"
05464             "# $4 = observer latitude [deg]\n"
05465             "# $5 = view point altitude [km]\n"
05466             "# $6 = view point longitude [deg]\n"
05467             "# $7 = view point latitude [deg]\n"
05468             "# $8 = tangent point altitude [km]\n"
05469             "# $9 = tangent point longitude [deg]\n"
05470             "# $10 = tangent point latitude [deg]\n");
05471     for (int id = 0; id < ctl->nd; id++)
05472         fprintf(out, "# $%d = radiance (%.4f cm^-1) [W/(m^2 sr cm^-1)]\n",
05473                 ++n, ctl->nu[id]);
05474     for (int id = 0; id < ctl->nd; id++)
05475         fprintf(out, "# $%d = transmittance (%.4f cm^-1) [-]\n", ++n,
05476                 ctl->nu[id]);
05477
05478     /* Write data... */
05479     for (int ir = 0; ir < obs->nr; ir++) {
05480         if (ir == 0 || obs->time[ir] != obs->time[ir - 1])
05481             fprintf(out, "\n");
05482         fprintf(out, "%.2f %g %g %g %g %g %g %g %g", obs->time[ir],
05483                 obs->obsz[ir], obs->obslon[ir], obs->obslat[ir],
05484                 obs->vpz[ir], obs->vplon[ir], obs->vplat[ir],
05485                 obs->tpz[ir], obs->tplon[ir], obs->tplat[ir]);
05486         for (int id = 0; id < ctl->nd; id++)
05487             fprintf(out, " %g", obs->rad[id][ir]);
05488         for (int id = 0; id < ctl->nd; id++)
05489             fprintf(out, " %g", obs->tau[id][ir]);
05490         fprintf(out, "\n");
05491     }
05492
05493     /* Close file... */
05494     fclose(out);
05495 }

```

**5.17.2.53 write\_shape()** void write\_shape (  
     const char \* filename,  
     double \* x,  
     double \* y,  
     int n )

Write shape function.

Definition at line 5499 of file [jurassic.c](#).

```

05503     {
05504
05505     FILE *out;
05506
05507     /* Write info... */
05508     LOG(1, "Write shape function: %s", filename);
05509
05510     /* Create file... */
05511     if (!(out = fopen(filename, "w")))
05512         ERRMSG("Cannot create file!");
05513
05514     /* Write header... */

```

```

05515     fprintf(out,
05516             "# $1 = shape function x-value [-]\n"
05517             "# $2 = shape function y-value [-]\n\n");
05518
05519     /* Write data... */
05520     for (int i = 0; i < n; i++)
05521         fprintf(out, "%.10g %.10g\n", x[i], y[i]);
05522
05523     /* Close file... */
05524     fclose(out);
05525 }

```

**5.17.2.54 write\_tbl()** void write\_tbl (

```

    ctl_t * ctl,
    tbl_t * tbl )

```

Write look-up table data.

Definition at line 5529 of file [jurassic.c](#).

```

05531     {
05532
05533     FILE *out;
05534
05535     char filename[2 * LEN];
05536
05537     /* Loop over emitters and detectors... */
05538     for (int ig = 0; ig < ctl->ng; ig++)
05539         for (int id = 0; id < ctl->nd; id++) {
05540
05541             /* Set filename... */
05542             sprintf(filename, "%s_%.4f_%.s.s", ctl->tblbase,
05543                     ctl->nu[id], ctl->emitter[ig],
05544                     ctl->tblfmt == 1 ? "tab" : "bin");
05545
05546             /* Write info... */
05547             LOG(1, "Write emissivity table: %s", filename);
05548
05549             /* Create file... */
05550             if (!(out = fopen(filename, "w")))
05551                 ERRMSG("Cannot create file!");
05552
05553             /* Write ASCII data... */
05554             if (ctl->tblfmt == 1) {
05555
05556                 /* Write header... */
05557                 fprintf(out,
05558                         "# $1 = pressure [hPa]\n"
05559                         "# $2 = temperature [K]\n"
05560                         "# $3 = column density [molecules/cm^2]\n"
05561                         "# $4 = emissivity [-]\n");
05562
05563                 /* Save table file... */
05564                 for (int ip = 0; ip < tbl->np[id][ig]; ip++)
05565                     for (int it = 0; it < tbl->nt[id][ig][ip]; it++) {
05566                         fprintf(out, "\n");
05567                         for (int iu = 0; iu < tbl->nu[id][ig][ip][it]; iu++)
05568                             fprintf(out, "%g %g %e %e\n",
05569                                     tbl->p[id][ig][ip], tbl->t[id][ig][ip][it],
05570                                     tbl->u[id][ig][ip][it][iu],
05571                                     tbl->eps[id][ig][ip][it][iu]);
05572                     }
05573             }
05574
05575             /* Write binary data... */
05576             else if (ctl->tblfmt == 2) {
05577                 FWRITE(&tbl->np[id][ig], int,
05578                        1,
05579                        out);
05580                 FWRITE(tbl->p[id][ig], double,
05581                        (size_t) tbl->np[id][ig],
05582                        out);
05583                 for (int ip = 0; ip < tbl->np[id][ig]; ip++) {
05584                     FWRITE(&tbl->nt[id][ig][ip], int,
05585                            1,
05586                            out);
05587                     FWRITE(tbl->t[id][ig][ip], double,
05588                            (size_t) tbl->nt[id][ig][ip],
05589                            out);

```

```

05590         for (int it = 0; it < tbl->nt[id][ig][ip]; it++) {
05591             FWRITE(&tbl->nu[id][ig][ip][it], int,
05592                 1,
05593                 out);
05594             FWRITE(tbl->u[id][ig][ip][it], float,
05595                 (size_t) tbl->nu[id][ig][ip][it],
05596                 out);
05597             FWRITE(tbl->eps[id][ig][ip][it], float,
05598                 (size_t) tbl->nu[id][ig][ip][it],
05599                 out);
05600         }
05601     }
05602 }
05603
05604 /* Error message... */
05605 else
05606     ERRMSG("Unknown look-up table format!");
05607
05608 /* Close file... */
05609 fclose(out);
05610 }
05611 }

```

### 5.17.2.55 x2atm() void x2atm (

```

    ctl_t * ctl,
    gsl_vector * x,
    atm_t * atm )

```

Decompose parameter vector or state vector.

Definition at line 5615 of file [jurassic.c](#).

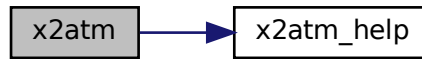
```

05618     {
05619
05620         size_t n = 0;
05621
05622         /* Get pressure... */
05623         for (int ip = 0; ip < atm->np; ip++)
05624             if (atm->z[ip] >= ctl->retp_zmin && atm->z[ip] <= ctl->retp_zmax)
05625                 x2atm_help(&atm->p[ip], x, &n);
05626
05627         /* Get temperature... */
05628         for (int ip = 0; ip < atm->np; ip++)
05629             if (atm->z[ip] >= ctl->rett_zmin && atm->z[ip] <= ctl->rett_zmax)
05630                 x2atm_help(&atm->t[ip], x, &n);
05631
05632         /* Get volume mixing ratio... */
05633         for (int ig = 0; ig < ctl->ng; ig++)
05634             for (int ip = 0; ip < atm->np; ip++)
05635                 if (atm->z[ip] >= ctl->retq_zmin[ig]
05636                     && atm->z[ip] <= ctl->retq_zmax[ig])
05637                     x2atm_help(&atm->q[ig][ip], x, &n);
05638
05639         /* Get extinction... */
05640         for (int iw = 0; iw < ctl->nw; iw++)
05641             for (int ip = 0; ip < atm->np; ip++)
05642                 if (atm->z[ip] >= ctl->retk_zmin[iw]
05643                     && atm->z[ip] <= ctl->retk_zmax[iw])
05644                     x2atm_help(&atm->k[iw][ip], x, &n);
05645
05646         /* Get cloud data... */
05647         if (ctl->ret_clz)
05648             x2atm_help(&atm->clz, x, &n);
05649         if (ctl->ret_cldz)
05650             x2atm_help(&atm->cldz, x, &n);
05651         if (ctl->ret_clk)
05652             for (int icl = 0; icl < ctl->ncl; icl++)
05653                 x2atm_help(&atm->clk[icl], x, &n);
05654
05655         /* Get surface data... */
05656         if (ctl->ret_sfz)
05657             x2atm_help(&atm->sfz, x, &n);
05658         if (ctl->ret_sfp)
05659             x2atm_help(&atm->sfp, x, &n);
05660         if (ctl->ret_sft)
05661             x2atm_help(&atm->sft, x, &n);
05662         if (ctl->ret_sfeps)
05663             for (int isf = 0; isf < ctl->nsf; isf++)
05664                 x2atm_help(&atm->sfeps[isf], x, &n);

```

```
05665 }
```

Here is the call graph for this function:



**5.17.2.56 x2atm\_help()** `void x2atm_help (`  
    `double * value,`  
    `gsl_vector * x,`  
    `size_t * n )`

Get element from state vector.

Definition at line 5669 of file [jurassic.c](#).

```
05672     {  
05673  
05674     /* Get state vector element... */  
05675     *value = gsl_vector_get(x, *n);  
05676     (*n)++;  
05677 }
```

**5.17.2.57 y2obs()** `void y2obs (`  
    `ctl_t * ctl,`  
    `gsl_vector * y,`  
    `obs_t * obs )`

Decompose measurement vector.

Definition at line 5681 of file [jurassic.c](#).

```
05684     {  
05685  
05686     size_t m = 0;  
05687  
05688     /* Decompose measurement vector... */  
05689     for (int ir = 0; ir < obs->nr; ir++)  
05690         for (int id = 0; id < ctl->nd; id++)  
05691             if (gsl_finite(obs->rad[id][ir])) {  
05692                 obs->rad[id][ir] = gsl_vector_get(y, m);  
05693                 m++;  
05694             }  
05695 }
```



## 5.18 jurassic.c

```

00001 /*
00002   This file is part of JURASSIC.
00003
00004   JURASSIC is free software: you can redistribute it and/or modify
00005   it under the terms of the GNU General Public License as published by
00006   the Free Software Foundation, either version 3 of the License, or
00007   (at your option) any later version.
00008
00009   JURASSIC is distributed in the hope that it will be useful,
00010   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012   GNU General Public License for more details.
00013
00014   You should have received a copy of the GNU eneral Public License
00015   along with JURASSIC. If not, see <http://www.gnu.org/licenses/>.
00016
00017   Copyright (C) 2003-2021 Forschungszentrum Juelich GmbH
00018 */
00019
00025 #include "jurassic.h"
00026
00027 /*****
00028
00029 size_t atm2x(
00030     ctl_t * ctl,
00031     atm_t * atm,
00032     gsl_vector * x,
00033     int *iqa,
00034     int *ipa) {
00035
00036     size_t n = 0;
00037
00038     /* Add pressure... */
00039     for (int ip = 0; ip < atm->np; ip++)
00040         if (atm->z[ip] >= ctl->retp_zmin && atm->z[ip] <= ctl->retp_zmax)
00041             atm2x_help(atm->p[ip], IDXP, ip, x, iqa, ipa, &n);
00042
00043     /* Add temperature... */
00044     for (int ip = 0; ip < atm->np; ip++)
00045         if (atm->z[ip] >= ctl->rett_zmin && atm->z[ip] <= ctl->rett_zmax)
00046             atm2x_help(atm->t[ip], IDXT, ip, x, iqa, ipa, &n);
00047
00048     /* Add volume mixing ratios... */
00049     for (int ig = 0; ig < ctl->ng; ig++)
00050         for (int ip = 0; ip < atm->np; ip++)
00051             if (atm->z[ip] >= ctl->retq_zmin[ig]
00052                 && atm->z[ip] <= ctl->retq_zmax[ig])
00053                 atm2x_help(atm->q[ig][ip], IDXQ(ig), ip, x, iqa, ipa, &n);
00054
00055     /* Add extinction... */
00056     for (int iw = 0; iw < ctl->nw; iw++)
00057         for (int ip = 0; ip < atm->np; ip++)
00058             if (atm->z[ip] >= ctl->retk_zmin[iw]
00059                 && atm->z[ip] <= ctl->retk_zmax[iw])
00060                 atm2x_help(atm->k[iw][ip], IDXK(iw), ip, x, iqa, ipa, &n);
00061
00062     /* Add cloud parameters... */
00063     if (ctl->ret_clz)
00064         atm2x_help(atm->clz, IDXCLZ, 0, x, iqa, ipa, &n);
00065     if (ctl->ret_cldz)
00066         atm2x_help(atm->cldz, IDXCLDZ, 0, x, iqa, ipa, &n);
00067     if (ctl->ret_clk)
00068         for (int icl = 0; icl < ctl->ncl; icl++)
00069             atm2x_help(atm->clk[icl], IDXCLK(icl), 0, x, iqa, ipa, &n);
00070
00071     /* Add surface parameters... */
00072     if (ctl->ret_sfz)
00073         atm2x_help(atm->sfz, IDXSFZ, 0, x, iqa, ipa, &n);
00074     if (ctl->ret_sfp)
00075         atm2x_help(atm->sfp, IDXSFP, 0, x, iqa, ipa, &n);
00076     if (ctl->ret_sft)
00077         atm2x_help(atm->sft, IDXSFT, 0, x, iqa, ipa, &n);
00078     if (ctl->ret_sfeps)
00079         for (int isf = 0; isf < ctl->nsf; isf++)
00080             atm2x_help(atm->sfeps[isf], IDXSFEPS(isf), 0, x, iqa, ipa, &n);
00081
00082     return n;
00083 }
00084
00085 /*****
00086
00087 void atm2x_help(
00088     double value,
00089     int value_iqa,
00090     int value_ip,

```

```

00091    gsl_vector * x,
00092    int *iga,
00093    int *ipa,
00094    size_t *n) {
00095
00096    /* Add element to state vector... */
00097    if (x != NULL)
00098        gsl_vector_set(x, *n, value);
00099    if (iga != NULL)
00100        iga[*n] = value_iga;
00101    if (ipa != NULL)
00102        ipa[*n] = value_ip;
00103    (*n)++;
00104 }
00105
00106
00107 /*****
00108
00109 double brightness(
00110     double rad,
00111     double nu) {
00112
00113     return C2 * nu / gsl_loglp(C1 * POW3(nu) / rad);
00114 }
00115
00116
00117 /*****
00118
00119 void cart2geo(
00120     double *x,
00121     double *z,
00122     double *lon,
00123     double *lat) {
00124
00125     double radius = NORM(x);
00126
00127     *lat = asin(x[2] / radius) * 180 / M_PI;
00128     *lon = atan2(x[1], x[0]) * 180 / M_PI;
00129     *z = radius - RE;
00130 }
00131
00132 /*****
00133
00134 void climatology(
00135     ctl_t * ctl,
00136     atm_t * atm) {
00137
00138     static double z[121] = {
00139         0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
00140         20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37,
00141         38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55,
00142         56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73,
00143         74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91,
00144         92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107,
00145         108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120
00146     };
00147
00148     static double pre[121] = {
00149         1017, 901.083, 796.45, 702.227, 617.614, 541.644, 473.437, 412.288,
00150         357.603, 308.96, 265.994, 228.348, 195.619, 167.351, 143.039, 122.198,
00151         104.369, 89.141, 76.1528, 65.0804, 55.641, 47.591, 40.7233, 34.8637,
00152         29.8633, 25.5956, 21.9534, 18.8445, 16.1909, 13.9258, 11.9913,
00153         10.34, 8.92988, 7.72454, 6.6924, 5.80701, 5.04654, 4.39238, 3.82902,
00154         3.34337, 2.92413, 2.56128, 2.2464, 1.97258, 1.73384, 1.52519, 1.34242,
00155         1.18197, 1.04086, 0.916546, 0.806832, 0.709875, 0.624101, 0.548176,
00156         0.480974, 0.421507, 0.368904, 0.322408, 0.281386, 0.245249, 0.213465,
00157         0.185549, 0.161072, 0.139644, 0.120913, 0.104568, 0.0903249, 0.0779269,
00158         0.0671493, 0.0577962, 0.0496902, 0.0426736, 0.0366093, 0.0313743,
00159         0.0268598, 0.0229699, 0.0196206, 0.0167399, 0.0142646, 0.0121397,
00160         0.0103181, 0.00875775, 0.00742226, 0.00628076, 0.00530519, 0.00447183,
00161         0.00376124, 0.00315632, 0.00264248, 0.00220738, 0.00184003, 0.00153095,
00162         0.00127204, 0.00105608, 0.000876652, 0.00072798, 0.00060492,
00163         0.000503201, 0.000419226, 0.000349896, 0.000292659, 0.000245421,
00164         0.000206394, 0.000174125, 0.000147441, 0.000125333, 0.000106985,
00165         9.173e-05, 7.90172e-05, 6.84172e-05, 5.95574e-05, 5.21183e-05,
00166         4.58348e-05, 4.05127e-05, 3.59987e-05, 3.21583e-05, 2.88718e-05,
00167         2.60322e-05, 2.35687e-05, 2.14263e-05, 1.95489e-05
00168     };
00169
00170     static double tem[121] = {
00171         285.14, 279.34, 273.91, 268.3, 263.24, 256.55, 250.2, 242.82, 236.17,
00172         229.87, 225.04, 221.19, 218.85, 217.19, 216.2, 215.68, 215.42, 215.55,
00173         215.92, 216.4, 216.93, 217.45, 218, 218.68, 219.39, 220.25, 221.3,
00174         222.41, 223.88, 225.42, 227.2, 229.52, 231.89, 234.51, 236.85, 239.42,
00175         241.94, 244.57, 247.36, 250.32, 253.34, 255.82, 258.27, 260.39,
00176         262.03, 263.45, 264.2, 264.78, 264.67, 264.38, 263.24, 262.03, 260.02,
00177         258.09, 255.63, 253.28, 250.43, 247.81, 245.26, 242.77, 240.38,

```

```
00178     237.94, 235.79, 233.53, 231.5, 229.53, 227.6, 225.62, 223.77, 222.06,
00179     220.33, 218.69, 217.18, 215.64, 214.13, 212.52, 210.86, 209.25,
00180     207.49, 205.81, 204.11, 202.22, 200.32, 198.39, 195.92, 193.46,
00181     190.94, 188.31, 185.82, 183.57, 181.43, 179.74, 178.64, 178.1, 178.25,
00182     178.7, 179.41, 180.67, 182.31, 184.18, 186.6, 189.53, 192.66, 196.54,
00183     201.13, 205.93, 211.73, 217.86, 225, 233.53, 242.57, 252.14, 261.48,
00184     272.97, 285.26, 299.12, 312.2, 324.17, 338.34, 352.56, 365.28
00185 };
00186
00187 static double c2h2[121] = {
00188     1.352e-09, 2.83e-10, 1.269e-10, 6.926e-11, 4.346e-11, 2.909e-11,
00189     2.014e-11, 1.363e-11, 8.71e-12, 5.237e-12, 2.718e-12, 1.375e-12,
00190     5.786e-13, 2.16e-13, 7.317e-14, 2.551e-14, 1.055e-14, 4.758e-15,
00191     2.056e-15, 7.703e-16, 2.82e-16, 1.035e-16, 4.382e-17, 1.946e-17,
00192     9.638e-18, 5.2e-18, 2.811e-18, 1.494e-18, 7.925e-19, 4.213e-19,
00193     1.998e-19, 8.78e-20, 3.877e-20, 1.728e-20, 7.743e-21, 3.536e-21,
00194     1.623e-21, 7.508e-22, 3.508e-22, 1.65e-22, 7.837e-23, 3.733e-23,
00195     1.808e-23, 8.77e-24, 4.285e-24, 2.095e-24, 1.032e-24, 5.082e-25,
00196     2.506e-25, 1.236e-25, 6.088e-26, 2.996e-26, 1.465e-26, 0, 0, 0,
00197     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00198     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00199     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00200 };
00201
00202 static double c2h6[121] = {
00203     2.667e-09, 2.02e-09, 1.658e-09, 1.404e-09, 1.234e-09, 1.109e-09,
00204     1.012e-09, 9.262e-10, 8.472e-10, 7.71e-10, 6.932e-10, 6.216e-10,
00205     5.503e-10, 4.87e-10, 4.342e-10, 3.861e-10, 3.347e-10, 2.772e-10,
00206     2.209e-10, 1.672e-10, 1.197e-10, 8.536e-11, 5.783e-11, 3.846e-11,
00207     2.495e-11, 1.592e-11, 1.017e-11, 6.327e-12, 3.895e-12, 2.403e-12,
00208     1.416e-12, 8.101e-13, 4.649e-13, 2.686e-13, 1.557e-13, 9.14e-14,
00209     5.386e-14, 3.19e-14, 1.903e-14, 1.14e-14, 6.875e-15, 4.154e-15,
00210     2.538e-15, 1.553e-15, 9.548e-16, 5.872e-16, 3.63e-16, 2.244e-16,
00211     1.388e-16, 8.587e-17, 5.308e-17, 3.279e-17, 2.017e-17, 1.238e-17,
00212     7.542e-18, 4.585e-18, 2.776e-18, 1.671e-18, 9.985e-19, 5.937e-19,
00213     3.518e-19, 2.07e-19, 1.215e-19, 7.06e-20, 4.097e-20, 2.37e-20,
00214     1.363e-20, 7.802e-21, 4.441e-21, 2.523e-21, 1.424e-21, 8.015e-22,
00215     4.497e-22, 2.505e-22, 1.391e-22, 7.691e-23, 4.238e-23, 2.331e-23,
00216     1.274e-23, 6.929e-24, 3.752e-24, 2.02e-24, 1.083e-24, 5.774e-25,
00217     3.041e-25, 1.593e-25, 8.308e-26, 4.299e-26, 2.195e-26, 1.112e-26,
00218     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00219     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00220 };
00221
00222 static double ccl4[121] = {
00223     1.075e-10, 1.075e-10, 1.075e-10, 1.075e-10, 1.075e-10, 1.075e-10,
00224     1.075e-10, 1.075e-10, 1.075e-10, 1.06e-10, 1.024e-10, 9.69e-11,
00225     8.93e-11, 8.078e-11, 7.213e-11, 6.307e-11, 5.383e-11, 4.49e-11,
00226     3.609e-11, 2.705e-11, 1.935e-11, 1.385e-11, 8.35e-12, 5.485e-12,
00227     3.853e-12, 2.22e-12, 5.875e-13, 3.445e-13, 1.015e-13, 6.075e-14,
00228     4.383e-14, 2.692e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00229     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00230     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00231     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00232     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00233     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00234     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00235     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00236     1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00237     1e-14, 1e-14, 1e-14
00238 };
00239
00240 static double ch4[121] = {
00241     1.864e-06, 1.835e-06, 1.819e-06, 1.805e-06, 1.796e-06, 1.788e-06,
00242     1.782e-06, 1.776e-06, 1.769e-06, 1.761e-06, 1.749e-06, 1.734e-06,
00243     1.716e-06, 1.692e-06, 1.654e-06, 1.61e-06, 1.567e-06, 1.502e-06,
00244     1.433e-06, 1.371e-06, 1.323e-06, 1.277e-06, 1.232e-06, 1.188e-06,
00245     1.147e-06, 1.108e-06, 1.07e-06, 1.027e-06, 9.854e-07, 9.416e-07,
00246     8.933e-07, 8.478e-07, 7.988e-07, 7.515e-07, 7.07e-07, 6.64e-07,
00247     6.239e-07, 5.864e-07, 5.512e-07, 5.184e-07, 4.87e-07, 4.571e-07,
00248     4.296e-07, 4.04e-07, 3.802e-07, 3.578e-07, 3.383e-07, 3.203e-07,
00249     3.032e-07, 2.889e-07, 2.76e-07, 2.635e-07, 2.519e-07, 2.409e-07,
00250     2.302e-07, 2.219e-07, 2.144e-07, 2.071e-07, 1.999e-07, 1.93e-07,
00251     1.862e-07, 1.795e-07, 1.731e-07, 1.668e-07, 1.607e-07, 1.548e-07,
00252     1.49e-07, 1.434e-07, 1.38e-07, 1.328e-07, 1.277e-07, 1.227e-07,
00253     1.18e-07, 1.134e-07, 1.089e-07, 1.046e-07, 1.004e-07, 9.635e-08,
00254     9.245e-08, 8.867e-08, 8.502e-08, 8.15e-08, 7.809e-08, 7.48e-08,
00255     7.159e-08, 6.849e-08, 6.55e-08, 6.262e-08, 5.98e-08, 5.708e-08,
00256     5.448e-08, 5.194e-08, 4.951e-08, 4.72e-08, 4.5e-08, 4.291e-08,
00257     4.093e-08, 3.905e-08, 3.729e-08, 3.563e-08, 3.408e-08, 3.265e-08,
00258     3.128e-08, 2.996e-08, 2.87e-08, 2.76e-08, 2.657e-08, 2.558e-08,
00259     2.467e-08, 2.385e-08, 2.307e-08, 2.234e-08, 2.168e-08, 2.108e-08,
00260     2.05e-08, 1.998e-08, 1.947e-08, 1.902e-08, 1.86e-08, 1.819e-08,
00261     1.782e-08
00262 };
00263
00264 static double clo[121] = {
```

```
00265     7.419e-15, 1.061e-14, 1.518e-14, 2.195e-14, 3.175e-14, 4.666e-14,
00266     6.872e-14, 1.03e-13, 1.553e-13, 2.375e-13, 3.664e-13, 5.684e-13,
00267     8.915e-13, 1.402e-12, 2.269e-12, 4.125e-12, 7.501e-12, 1.257e-11,
00268     2.048e-11, 3.338e-11, 5.44e-11, 8.846e-11, 1.008e-10, 1.082e-10,
00269     1.157e-10, 1.232e-10, 1.312e-10, 1.539e-10, 1.822e-10, 2.118e-10,
00270     2.387e-10, 2.687e-10, 2.875e-10, 3.031e-10, 3.23e-10, 3.648e-10,
00271     4.117e-10, 4.477e-10, 4.633e-10, 4.794e-10, 4.95e-10, 5.104e-10,
00272     5.259e-10, 5.062e-10, 4.742e-10, 4.443e-10, 4.051e-10, 3.659e-10,
00273     3.305e-10, 2.911e-10, 2.54e-10, 2.215e-10, 1.927e-10, 1.675e-10,
00274     1.452e-10, 1.259e-10, 1.09e-10, 9.416e-11, 8.119e-11, 6.991e-11,
00275     6.015e-11, 5.163e-11, 4.43e-11, 3.789e-11, 3.24e-11, 2.769e-11,
00276     2.361e-11, 2.011e-11, 1.71e-11, 1.453e-11, 1.233e-11, 1.045e-11,
00277     8.851e-12, 7.48e-12, 6.316e-12, 5.326e-12, 4.487e-12, 3.778e-12,
00278     3.176e-12, 2.665e-12, 2.234e-12, 1.87e-12, 1.563e-12, 1.304e-12,
00279     1.085e-12, 9.007e-13, 7.468e-13, 6.179e-13, 5.092e-13, 4.188e-13,
00280     3.442e-13, 2.816e-13, 2.304e-13, 1.885e-13, 1.542e-13, 1.263e-13,
00281     1.035e-13, 8.5e-14, 7.004e-14, 5.783e-14, 4.795e-14, 4.007e-14,
00282     3.345e-14, 2.792e-14, 2.33e-14, 1.978e-14, 1.686e-14, 1.438e-14,
00283     1.234e-14, 1.07e-14, 9.312e-15, 8.131e-15, 7.164e-15, 6.367e-15,
00284     5.67e-15, 5.088e-15, 4.565e-15, 4.138e-15, 3.769e-15, 3.432e-15,
00285     3.148e-15
00286 };
00287
00288 static double clono2[121] = {
00289     1.011e-13, 1.515e-13, 2.272e-13, 3.446e-13, 5.231e-13, 8.085e-13,
00290     1.253e-12, 1.979e-12, 3.149e-12, 5.092e-12, 8.312e-12, 1.366e-11,
00291     2.272e-11, 3.791e-11, 6.209e-11, 9.101e-11, 1.334e-10, 1.951e-10,
00292     2.853e-10, 3.94e-10, 4.771e-10, 5.771e-10, 6.675e-10, 7.665e-10,
00293     8.504e-10, 8.924e-10, 9.363e-10, 8.923e-10, 8.411e-10, 7.646e-10,
00294     6.525e-10, 5.576e-10, 4.398e-10, 3.403e-10, 2.612e-10, 1.915e-10,
00295     1.407e-10, 1.028e-10, 7.455e-11, 5.42e-11, 3.708e-11, 2.438e-11,
00296     1.618e-11, 1.075e-11, 7.17e-12, 4.784e-12, 3.205e-12, 2.147e-12,
00297     1.44e-12, 9.654e-13, 6.469e-13, 4.332e-13, 2.891e-13, 1.926e-13,
00298     1.274e-13, 8.422e-14, 5.547e-14, 3.636e-14, 2.368e-14, 1.536e-14,
00299     9.937e-15, 6.39e-15, 4.101e-15, 2.61e-15, 1.659e-15, 1.052e-15,
00300     6.638e-16, 4.172e-16, 2.61e-16, 1.63e-16, 1.013e-16, 6.275e-17,
00301     3.879e-17, 2.383e-17, 1.461e-17, 8.918e-18, 5.43e-18, 3.301e-18,
00302     1.997e-18, 1.203e-18, 7.216e-19, 4.311e-19, 2.564e-19, 1.519e-19,
00303     8.911e-20, 5.203e-20, 3.026e-20, 1.748e-20, 9.99e-21, 5.673e-21,
00304     3.215e-21, 1.799e-21, 1.006e-21, 5.628e-22, 3.146e-22, 1.766e-22,
00305     9.94e-23, 5.614e-23, 3.206e-23, 1.841e-23, 1.071e-23, 6.366e-24,
00306     3.776e-24, 2.238e-24, 1.326e-24, 8.253e-25, 5.201e-25, 3.279e-25,
00307     2.108e-25, 1.395e-25, 9.326e-26, 6.299e-26, 4.365e-26, 3.104e-26,
00308     2.219e-26, 1.621e-26, 1.185e-26, 8.92e-27, 6.804e-27, 5.191e-27,
00309     4.041e-27
00310 };
00311
00312 static double co[121] = {
00313     1.907e-07, 1.553e-07, 1.362e-07, 1.216e-07, 1.114e-07, 1.036e-07,
00314     9.737e-08, 9.152e-08, 8.559e-08, 7.966e-08, 7.277e-08, 6.615e-08,
00315     5.884e-08, 5.22e-08, 4.699e-08, 4.284e-08, 3.776e-08, 3.274e-08,
00316     2.845e-08, 2.479e-08, 2.246e-08, 2.054e-08, 1.991e-08, 1.951e-08,
00317     1.94e-08, 2.009e-08, 2.1e-08, 2.201e-08, 2.322e-08, 2.45e-08,
00318     2.602e-08, 2.73e-08, 2.867e-08, 2.998e-08, 3.135e-08, 3.255e-08,
00319     3.352e-08, 3.426e-08, 3.484e-08, 3.53e-08, 3.593e-08, 3.671e-08,
00320     3.759e-08, 3.945e-08, 4.192e-08, 4.49e-08, 5.03e-08, 5.703e-08,
00321     6.538e-08, 7.878e-08, 9.644e-08, 1.196e-07, 1.498e-07, 1.904e-07,
00322     2.422e-07, 3.055e-07, 3.804e-07, 4.747e-07, 5.899e-07, 7.272e-07,
00323     8.91e-07, 1.071e-06, 1.296e-06, 1.546e-06, 1.823e-06, 2.135e-06,
00324     2.44e-06, 2.714e-06, 2.967e-06, 3.189e-06, 3.391e-06, 3.58e-06,
00325     3.773e-06, 4.022e-06, 4.346e-06, 4.749e-06, 5.199e-06, 5.668e-06,
00326     6.157e-06, 6.688e-06, 7.254e-06, 7.867e-06, 8.539e-06, 9.26e-06,
00327     1.009e-05, 1.119e-05, 1.228e-05, 1.365e-05, 1.506e-05, 1.641e-05,
00328     1.784e-05, 1.952e-05, 2.132e-05, 2.323e-05, 2.531e-05, 2.754e-05,
00329     3.047e-05, 3.459e-05, 3.922e-05, 4.439e-05, 4.825e-05, 5.077e-05,
00330     5.34e-05, 5.618e-05, 5.909e-05, 6.207e-05, 6.519e-05, 6.845e-05,
00331     6.819e-05, 6.726e-05, 6.622e-05, 6.512e-05, 6.671e-05, 6.862e-05,
00332     7.048e-05, 7.264e-05, 7.3e-05, 7.3e-05, 7.3e-05, 7.3e-05, 7.3e-05
00333 };
00334
00335 static double cof2[121] = {
00336     7.5e-14, 1.055e-13, 1.485e-13, 2.111e-13, 3.001e-13, 4.333e-13,
00337     6.269e-13, 9.221e-13, 1.364e-12, 2.046e-12, 3.093e-12, 4.703e-12,
00338     7.225e-12, 1.113e-11, 1.66e-11, 2.088e-11, 2.626e-11, 3.433e-11,
00339     4.549e-11, 5.886e-11, 7.21e-11, 8.824e-11, 1.015e-10, 1.155e-10,
00340     1.288e-10, 1.388e-10, 1.497e-10, 1.554e-10, 1.606e-10, 1.639e-10,
00341     1.64e-10, 1.64e-10, 1.596e-10, 1.542e-10, 1.482e-10, 1.382e-10,
00342     1.289e-10, 1.198e-10, 1.109e-10, 1.026e-10, 9.484e-11, 8.75e-11,
00343     8.086e-11, 7.49e-11, 6.948e-11, 6.446e-11, 5.961e-11, 5.505e-11,
00344     5.085e-11, 4.586e-11, 4.1e-11, 3.665e-11, 3.235e-11, 2.842e-11,
00345     2.491e-11, 2.11e-11, 1.769e-11, 1.479e-11, 1.197e-11, 9.631e-12,
00346     7.74e-12, 6.201e-12, 4.963e-12, 3.956e-12, 3.151e-12, 2.507e-12,
00347     1.99e-12, 1.576e-12, 1.245e-12, 9.83e-13, 7.742e-13, 6.088e-13,
00348     4.782e-13, 3.745e-13, 2.929e-13, 2.286e-13, 1.782e-13, 1.388e-13,
00349     1.079e-13, 8.362e-14, 6.471e-14, 4.996e-14, 3.85e-14, 2.96e-14,
00350     2.265e-14, 1.729e-14, 1.317e-14, 9.998e-15, 7.549e-15, 5.683e-15,
00351     4.273e-15, 3.193e-15, 2.385e-15, 1.782e-15, 1.331e-15, 9.957e-16,
```

```
00352     7.461e-16, 5.601e-16, 4.228e-16, 3.201e-16, 2.438e-16, 1.878e-16,
00353     1.445e-16, 1.111e-16, 8.544e-17, 6.734e-17, 5.341e-17, 4.237e-17,
00354     3.394e-17, 2.759e-17, 2.254e-17, 1.851e-17, 1.54e-17, 1.297e-17,
00355     1.096e-17, 9.365e-18, 8e-18, 6.938e-18, 6.056e-18, 5.287e-18,
00356     4.662e-18
00357 };
00358
00359 static double f11[121] = {
00360     2.65e-10, 2.65e-10, 2.65e-10, 2.65e-10, 2.65e-10, 2.65e-10, 2.65e-10,
00361     2.65e-10, 2.65e-10, 2.65e-10, 2.65e-10, 2.65e-10, 2.635e-10, 2.536e-10,
00362     2.44e-10, 2.348e-10, 2.258e-10, 2.153e-10, 2.046e-10, 1.929e-10,
00363     1.782e-10, 1.648e-10, 1.463e-10, 1.291e-10, 1.1e-10, 8.874e-11,
00364     7.165e-11, 5.201e-11, 3.744e-11, 2.577e-11, 1.64e-11, 1.048e-11,
00365     5.993e-12, 3.345e-12, 1.839e-12, 9.264e-13, 4.688e-13, 2.329e-13,
00366     1.129e-13, 5.505e-14, 2.825e-14, 1.492e-14, 7.997e-15, 5.384e-15,
00367     3.988e-15, 2.955e-15, 2.196e-15, 1.632e-15, 1.214e-15, 9.025e-16,
00368     6.708e-16, 4.984e-16, 3.693e-16, 2.733e-16, 2.013e-16, 1.481e-16,
00369     1.087e-16, 7.945e-17, 5.782e-17, 4.195e-17, 3.038e-17, 2.19e-17,
00370     1.577e-17, 1.128e-17, 8.063e-18, 5.753e-18, 4.09e-18, 2.899e-18,
00371     2.048e-18, 1.444e-18, 1.015e-18, 7.12e-19, 4.985e-19, 3.474e-19,
00372     2.417e-19, 1.677e-19, 1.161e-19, 8.029e-20, 5.533e-20, 3.799e-20,
00373     2.602e-20, 1.776e-20, 1.209e-20, 8.202e-21, 5.522e-21, 3.707e-21,
00374     2.48e-21, 1.652e-21, 1.091e-21, 7.174e-22, 4.709e-22, 3.063e-22,
00375     1.991e-22, 1.294e-22, 8.412e-23, 5.483e-23, 3.581e-23, 2.345e-23,
00376     1.548e-23, 1.027e-23, 6.869e-24, 4.673e-24, 3.173e-24, 2.153e-24,
00377     1.461e-24, 1.028e-24, 7.302e-25, 5.188e-25, 3.739e-25, 2.753e-25,
00378     2.043e-25, 1.528e-25, 1.164e-25, 9.041e-26, 7.051e-26, 5.587e-26,
00379     4.428e-26, 3.588e-26, 2.936e-26, 2.402e-26, 1.995e-26
00380 };
00381
00382 static double f12[121] = {
00383     5.45e-10, 5.45e-10, 5.45e-10, 5.45e-10, 5.45e-10, 5.45e-10, 5.45e-10,
00384     5.45e-10, 5.45e-10, 5.45e-10, 5.45e-10, 5.45e-10, 5.429e-10, 5.291e-10,
00385     5.155e-10, 5.022e-10, 4.893e-10, 4.772e-10, 4.655e-10, 4.497e-10,
00386     4.249e-10, 4.015e-10, 3.632e-10, 3.261e-10, 2.858e-10, 2.408e-10,
00387     2.03e-10, 1.685e-10, 1.4e-10, 1.163e-10, 9.65e-11, 8.02e-11, 6.705e-11,
00388     5.624e-11, 4.764e-11, 4.249e-11, 3.792e-11, 3.315e-11, 2.819e-11,
00389     2.4e-11, 1.999e-11, 1.64e-11, 1.352e-11, 1.14e-11, 9.714e-12,
00390     8.28e-12, 7.176e-12, 6.251e-12, 5.446e-12, 4.72e-12, 4.081e-12,
00391     3.528e-12, 3.08e-12, 2.699e-12, 2.359e-12, 2.111e-12, 1.901e-12,
00392     1.709e-12, 1.534e-12, 1.376e-12, 1.233e-12, 1.103e-12, 9.869e-13,
00393     8.808e-13, 7.859e-13, 7.008e-13, 6.241e-13, 5.553e-13, 4.935e-13,
00394     4.383e-13, 3.889e-13, 3.447e-13, 3.054e-13, 2.702e-13, 2.389e-13,
00395     2.11e-13, 1.862e-13, 1.643e-13, 1.448e-13, 1.274e-13, 1.121e-13,
00396     9.844e-14, 8.638e-14, 7.572e-14, 6.62e-14, 5.782e-14, 5.045e-14,
00397     4.394e-14, 3.817e-14, 3.311e-14, 2.87e-14, 2.48e-14, 2.142e-14,
00398     1.851e-14, 1.599e-14, 1.383e-14, 1.196e-14, 1.036e-14, 9e-15,
00399     7.828e-15, 6.829e-15, 5.992e-15, 5.254e-15, 4.606e-15, 4.037e-15,
00400     3.583e-15, 3.19e-15, 2.841e-15, 2.542e-15, 2.291e-15, 2.07e-15,
00401     1.875e-15, 1.71e-15, 1.57e-15, 1.442e-15, 1.333e-15, 1.232e-15,
00402     1.147e-15, 1.071e-15, 1.001e-15, 9.396e-16
00403 };
00404
00405 static double f14[121] = {
00406     9e-11, 9e-11, 9e-11, 9e-11, 9e-11, 9e-11, 9e-11, 9e-11, 9e-11, 9e-11,
00407     9e-11, 9e-11, 9e-11, 9e-11, 9e-11, 8.91e-11, 8.73e-11, 8.46e-11,
00408     8.19e-11, 7.92e-11, 7.74e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00409     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00410     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00411     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00412     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00413     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00414     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00415     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00416     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00417     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00418     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00419     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00420     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00421     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00422     7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11
00423 };
00424
00425 static double f22[121] = {
00426     1.4e-10, 1.4e-10, 1.4e-10, 1.4e-10, 1.4e-10, 1.4e-10, 1.4e-10,
00427     1.4e-10, 1.4e-10, 1.4e-10, 1.372e-10, 1.317e-10, 1.235e-10, 1.153e-10,
00428     1.075e-10, 1.002e-10, 9.332e-11, 8.738e-11, 8.194e-11, 7.7e-11,
00429     7.165e-11, 6.753e-11, 6.341e-11, 5.971e-11, 5.6e-11, 5.229e-11,
00430     4.859e-11, 4.488e-11, 4.118e-11, 3.83e-11, 3.568e-11, 3.308e-11,
00431     3.047e-11, 2.82e-11, 2.594e-11, 2.409e-11, 2.237e-11, 2.065e-11,
00432     1.894e-11, 1.771e-11, 1.647e-11, 1.532e-11, 1.416e-11, 1.332e-11,
00433     1.246e-11, 1.161e-11, 1.087e-11, 1.017e-11, 9.471e-12, 8.853e-12,
00434     8.235e-12, 7.741e-12, 7.247e-12, 6.836e-12, 6.506e-12, 6.176e-12,
00435     5.913e-12, 5.65e-12, 5.419e-12, 5.221e-12, 5.024e-12, 4.859e-12,
00436     4.694e-12, 4.546e-12, 4.414e-12, 4.282e-12, 4.15e-12, 4.019e-12,
00437     3.903e-12, 3.805e-12, 3.706e-12, 3.607e-12, 3.508e-12, 3.41e-12,
00438     3.31e-12, 3.212e-12, 3.129e-12, 3.047e-12, 2.964e-12, 2.882e-12,
```

```
00439     2.8e-12, 2.734e-12, 2.668e-12, 2.602e-12, 2.537e-12, 2.471e-12,
00440     2.421e-12, 2.372e-12, 2.322e-12, 2.273e-12, 2.224e-12, 2.182e-12,
00441     2.141e-12, 2.1e-12, 2.059e-12, 2.018e-12, 1.977e-12, 1.935e-12,
00442     1.894e-12, 1.853e-12, 1.812e-12, 1.77e-12, 1.73e-12, 1.688e-12,
00443     1.647e-12, 1.606e-12, 1.565e-12, 1.524e-12, 1.483e-12, 1.441e-12,
00444     1.4e-12, 1.359e-12, 1.317e-12, 1.276e-12, 1.235e-12, 1.194e-12,
00445     1.153e-12, 1.112e-12, 1.071e-12, 1.029e-12, 9.883e-13
00446 };
00447
00448 static double h2o[121] = {
00449     0.01166, 0.008269, 0.005742, 0.003845, 0.00277, 0.001897, 0.001272,
00450     0.000827, 0.000539, 0.0003469, 0.0001579, 3.134e-05, 1.341e-05,
00451     6.764e-06, 4.498e-06, 3.703e-06, 3.724e-06, 3.899e-06, 4.002e-06,
00452     4.122e-06, 4.277e-06, 4.438e-06, 4.558e-06, 4.673e-06, 4.763e-06,
00453     4.809e-06, 4.856e-06, 4.936e-06, 5.021e-06, 5.114e-06, 5.222e-06,
00454     5.331e-06, 5.414e-06, 5.488e-06, 5.563e-06, 5.633e-06, 5.704e-06,
00455     5.767e-06, 5.819e-06, 5.872e-06, 5.914e-06, 5.949e-06, 5.984e-06,
00456     6.015e-06, 6.044e-06, 6.073e-06, 6.104e-06, 6.136e-06, 6.167e-06,
00457     6.189e-06, 6.208e-06, 6.226e-06, 6.212e-06, 6.185e-06, 6.158e-06,
00458     6.114e-06, 6.066e-06, 6.018e-06, 5.877e-06, 5.728e-06, 5.582e-06,
00459     5.437e-06, 5.296e-06, 5.156e-06, 5.02e-06, 4.886e-06, 4.754e-06,
00460     4.625e-06, 4.498e-06, 4.374e-06, 4.242e-06, 4.096e-06, 3.955e-06,
00461     3.817e-06, 3.683e-06, 3.491e-06, 3.204e-06, 2.94e-06, 2.696e-06,
00462     2.47e-06, 2.252e-06, 2.019e-06, 1.808e-06, 1.618e-06, 1.445e-06,
00463     1.285e-06, 1.105e-06, 9.489e-07, 8.121e-07, 6.938e-07, 5.924e-07,
00464     5.04e-07, 4.288e-07, 3.648e-07, 3.103e-07, 2.642e-07, 2.252e-07,
00465     1.921e-07, 1.643e-07, 1.408e-07, 1.211e-07, 1.048e-07, 9.063e-08,
00466     7.835e-08, 6.774e-08, 5.936e-08, 5.221e-08, 4.592e-08, 4.061e-08,
00467     3.62e-08, 3.236e-08, 2.902e-08, 2.62e-08, 2.383e-08, 2.171e-08,
00468     1.989e-08, 1.823e-08, 1.684e-08, 1.562e-08, 1.449e-08, 1.351e-08
00469 };
00470
00471 static double h2o2[121] = {
00472     1.779e-10, 7.938e-10, 8.953e-10, 8.032e-10, 6.564e-10, 5.159e-10,
00473     4.003e-10, 3.026e-10, 2.222e-10, 1.58e-10, 1.044e-10, 6.605e-11,
00474     3.413e-11, 1.453e-11, 1.062e-11, 1.009e-11, 9.597e-12, 1.175e-11,
00475     1.572e-11, 2.091e-11, 2.746e-11, 3.603e-11, 4.791e-11, 6.387e-11,
00476     8.239e-11, 1.007e-10, 1.23e-10, 1.363e-10, 1.489e-10, 1.585e-10,
00477     1.608e-10, 1.632e-10, 1.576e-10, 1.502e-10, 1.423e-10, 1.302e-10,
00478     1.192e-10, 1.085e-10, 9.795e-11, 8.854e-11, 8.057e-11, 7.36e-11,
00479     6.736e-11, 6.362e-11, 6.087e-11, 5.825e-11, 5.623e-11, 5.443e-11,
00480     5.27e-11, 5.098e-11, 4.931e-11, 4.769e-11, 4.611e-11, 4.458e-11,
00481     4.308e-11, 4.102e-11, 3.887e-11, 3.682e-11, 3.521e-11, 3.369e-11,
00482     3.224e-11, 3.082e-11, 2.946e-11, 2.814e-11, 2.687e-11, 2.566e-11,
00483     2.449e-11, 2.336e-11, 2.227e-11, 2.123e-11, 2.023e-11, 1.927e-11,
00484     1.835e-11, 1.746e-11, 1.661e-11, 1.58e-11, 1.502e-11, 1.428e-11,
00485     1.357e-11, 1.289e-11, 1.224e-11, 1.161e-11, 1.102e-11, 1.045e-11,
00486     9.895e-12, 9.369e-12, 8.866e-12, 8.386e-12, 7.922e-12, 7.479e-12,
00487     7.06e-12, 6.656e-12, 6.274e-12, 5.914e-12, 5.575e-12, 5.257e-12,
00488     4.959e-12, 4.679e-12, 4.42e-12, 4.178e-12, 3.954e-12, 3.75e-12,
00489     3.557e-12, 3.372e-12, 3.198e-12, 3.047e-12, 2.908e-12, 2.775e-12,
00490     2.653e-12, 2.544e-12, 2.442e-12, 2.346e-12, 2.26e-12, 2.183e-12,
00491     2.11e-12, 2.044e-12, 1.98e-12, 1.924e-12, 1.871e-12, 1.821e-12,
00492     1.775e-12
00493 };
00494
00495 static double hcn[121] = {
00496     5.5e-10, 5.5e-10, 5.5e-10, 5.5e-10, 5.5e-10, 5.5e-10, 5.5e-10,
00497     5.5e-10, 5.5e-10, 5.5e-10, 5.5e-10, 5.498e-10, 5.495e-10, 5.493e-10,
00498     5.49e-10, 5.488e-10, 4.717e-10, 3.946e-10, 3.174e-10, 2.4e-10,
00499     1.626e-10, 1.619e-10, 1.612e-10, 1.602e-10, 1.593e-10, 1.582e-10,
00500     1.572e-10, 1.56e-10, 1.549e-10, 1.539e-10, 1.53e-10, 1.519e-10,
00501     1.506e-10, 1.487e-10, 1.467e-10, 1.449e-10, 1.43e-10, 1.413e-10,
00502     1.397e-10, 1.382e-10, 1.368e-10, 1.354e-10, 1.337e-10, 1.315e-10,
00503     1.292e-10, 1.267e-10, 1.241e-10, 1.215e-10, 1.19e-10, 1.165e-10,
00504     1.141e-10, 1.118e-10, 1.096e-10, 1.072e-10, 1.047e-10, 1.021e-10,
00505     9.968e-11, 9.739e-11, 9.539e-11, 9.339e-11, 9.135e-11, 8.898e-11,
00506     8.664e-11, 8.439e-11, 8.249e-11, 8.075e-11, 7.904e-11, 7.735e-11,
00507     7.565e-11, 7.399e-11, 7.245e-11, 7.109e-11, 6.982e-11, 6.863e-11,
00508     6.755e-11, 6.657e-11, 6.587e-11, 6.527e-11, 6.476e-11, 6.428e-11,
00509     6.382e-11, 6.343e-11, 6.307e-11, 6.272e-11, 6.238e-11, 6.205e-11,
00510     6.17e-11, 6.137e-11, 6.102e-11, 6.072e-11, 6.046e-11, 6.03e-11,
00511     6.018e-11, 6.01e-11, 6.001e-11, 5.992e-11, 5.984e-11, 5.975e-11,
00512     5.967e-11, 5.958e-11, 5.95e-11, 5.941e-11, 5.933e-11, 5.925e-11,
00513     5.916e-11, 5.908e-11, 5.899e-11, 5.891e-11, 5.883e-11, 5.874e-11,
00514     5.866e-11, 5.858e-11, 5.85e-11, 5.841e-11, 5.833e-11, 5.825e-11,
00515     5.817e-11, 5.808e-11, 5.8e-11, 5.792e-11, 5.784e-11
00516 };
00517
00518 static double hno3[121] = {
00519     1.809e-10, 7.234e-10, 5.899e-10, 4.342e-10, 3.277e-10, 2.661e-10,
00520     2.35e-10, 2.267e-10, 2.389e-10, 2.651e-10, 3.255e-10, 4.099e-10,
00521     5.42e-10, 6.978e-10, 8.807e-10, 1.112e-09, 1.405e-09, 2.04e-09,
00522     3.111e-09, 4.5e-09, 5.762e-09, 7.37e-09, 7.852e-09, 8.109e-09,
00523     8.067e-09, 7.554e-09, 7.076e-09, 6.268e-09, 5.524e-09, 4.749e-09,
00524     3.909e-09, 3.223e-09, 2.517e-09, 1.942e-09, 1.493e-09, 1.122e-09,
00525     8.449e-10, 6.361e-10, 4.787e-10, 3.611e-10, 2.804e-10, 2.215e-10,
```

```
00526    1.758e-10, 1.441e-10, 1.197e-10, 9.953e-11, 8.505e-11, 7.334e-11,
00527    6.325e-11, 5.625e-11, 5.058e-11, 4.548e-11, 4.122e-11, 3.748e-11,
00528    3.402e-11, 3.088e-11, 2.8e-11, 2.536e-11, 2.293e-11, 2.072e-11,
00529    1.871e-11, 1.687e-11, 1.52e-11, 1.368e-11, 1.23e-11, 1.105e-11,
00530    9.922e-12, 8.898e-12, 7.972e-12, 7.139e-12, 6.385e-12, 5.708e-12,
00531    5.099e-12, 4.549e-12, 4.056e-12, 3.613e-12, 3.216e-12, 2.862e-12,
00532    2.544e-12, 2.259e-12, 2.004e-12, 1.776e-12, 1.572e-12, 1.391e-12,
00533    1.227e-12, 1.082e-12, 9.528e-13, 8.379e-13, 7.349e-13, 6.436e-13,
00534    5.634e-13, 4.917e-13, 4.291e-13, 3.745e-13, 3.267e-13, 2.854e-13,
00535    2.494e-13, 2.181e-13, 1.913e-13, 1.68e-13, 1.479e-13, 1.31e-13,
00536    1.159e-13, 1.025e-13, 9.067e-14, 8.113e-14, 7.281e-14, 6.535e-14,
00537    5.892e-14, 5.348e-14, 4.867e-14, 4.439e-14, 4.073e-14, 3.76e-14,
00538    3.476e-14, 3.229e-14, 3e-14, 2.807e-14, 2.635e-14, 2.473e-14,
00539    2.332e-14
00540 };
00541
00542 static double hno4[121] = {
00543    6.118e-12, 3.594e-12, 2.807e-12, 3.04e-12, 4.458e-12, 7.986e-12,
00544    1.509e-11, 2.661e-11, 3.738e-11, 4.652e-11, 4.429e-11, 3.992e-11,
00545    3.347e-11, 3.005e-11, 3.173e-11, 4.055e-11, 5.812e-11, 8.489e-11,
00546    1.19e-10, 1.482e-10, 1.766e-10, 2.103e-10, 2.35e-10, 2.598e-10,
00547    2.801e-10, 2.899e-10, 3e-10, 2.817e-10, 2.617e-10, 2.332e-10,
00548    1.933e-10, 1.605e-10, 1.232e-10, 9.285e-11, 6.941e-11, 4.951e-11,
00549    3.539e-11, 2.402e-11, 1.522e-11, 9.676e-12, 6.056e-12, 3.745e-12,
00550    2.34e-12, 1.463e-12, 9.186e-13, 5.769e-13, 3.322e-13, 1.853e-13,
00551    1.035e-13, 7.173e-14, 5.382e-14, 4.036e-14, 3.401e-14, 2.997e-14,
00552    2.635e-14, 2.316e-14, 2.034e-14, 1.783e-14, 1.56e-14, 1.363e-14,
00553    1.19e-14, 1.037e-14, 9.032e-15, 7.846e-15, 6.813e-15, 5.912e-15,
00554    5.121e-15, 4.431e-15, 3.829e-15, 3.306e-15, 2.851e-15, 2.456e-15,
00555    2.114e-15, 1.816e-15, 1.559e-15, 1.337e-15, 1.146e-15, 9.811e-16,
00556    8.389e-16, 7.162e-16, 6.109e-16, 5.203e-16, 4.425e-16, 3.76e-16,
00557    3.184e-16, 2.692e-16, 2.274e-16, 1.917e-16, 1.61e-16, 1.35e-16,
00558    1.131e-16, 9.437e-17, 7.874e-17, 6.57e-17, 5.481e-17, 4.579e-17,
00559    3.828e-17, 3.204e-17, 2.691e-17, 2.264e-17, 1.912e-17, 1.626e-17,
00560    1.382e-17, 1.174e-17, 9.972e-18, 8.603e-18, 7.45e-18, 6.453e-18,
00561    5.623e-18, 4.944e-18, 4.361e-18, 3.859e-18, 3.443e-18, 3.096e-18,
00562    2.788e-18, 2.528e-18, 2.293e-18, 2.099e-18, 1.929e-18, 1.773e-18,
00563    1.64e-18
00564 };
00565
00566 static double hocl[121] = {
00567    1.056e-12, 1.194e-12, 1.35e-12, 1.531e-12, 1.737e-12, 1.982e-12,
00568    2.263e-12, 2.599e-12, 2.991e-12, 3.459e-12, 4.012e-12, 4.662e-12,
00569    5.438e-12, 6.35e-12, 7.425e-12, 8.686e-12, 1.016e-11, 1.188e-11,
00570    1.389e-11, 1.659e-11, 2.087e-11, 2.621e-11, 3.265e-11, 4.064e-11,
00571    4.859e-11, 5.441e-11, 6.09e-11, 6.373e-11, 6.611e-11, 6.94e-11,
00572    7.44e-11, 7.97e-11, 8.775e-11, 9.722e-11, 1.064e-10, 1.089e-10,
00573    1.114e-10, 1.106e-10, 1.053e-10, 1.004e-10, 9.006e-11, 7.778e-11,
00574    6.739e-11, 5.636e-11, 4.655e-11, 3.845e-11, 3.042e-11, 2.368e-11,
00575    1.845e-11, 1.442e-11, 1.127e-11, 8.814e-12, 6.544e-12, 4.763e-12,
00576    3.449e-12, 2.612e-12, 1.999e-12, 1.526e-12, 1.16e-12, 8.793e-13,
00577    6.655e-13, 5.017e-13, 3.778e-13, 2.829e-13, 2.117e-13, 1.582e-13,
00578    1.178e-13, 8.755e-14, 6.486e-14, 4.799e-14, 3.54e-14, 2.606e-14,
00579    1.916e-14, 1.403e-14, 1.026e-14, 7.48e-15, 5.446e-15, 3.961e-15,
00580    2.872e-15, 2.076e-15, 1.498e-15, 1.077e-15, 7.726e-16, 5.528e-16,
00581    3.929e-16, 2.785e-16, 1.969e-16, 1.386e-16, 9.69e-17, 6.747e-17,
00582    4.692e-17, 3.236e-17, 2.232e-17, 1.539e-17, 1.061e-17, 7.332e-18,
00583    5.076e-18, 3.522e-18, 2.461e-18, 1.726e-18, 1.22e-18, 8.75e-19,
00584    6.264e-19, 4.482e-19, 3.207e-19, 2.368e-19, 1.762e-19, 1.312e-19,
00585    9.891e-20, 7.595e-20, 5.87e-20, 4.567e-20, 3.612e-20, 2.904e-20,
00586    2.343e-20, 1.917e-20, 1.568e-20, 1.308e-20, 1.1e-20, 9.25e-21,
00587    7.881e-21
00588 };
00589
00590 static double n2o[121] = {
00591    3.17e-07, 3.17e-07, 3.17e-07, 3.17e-07, 3.17e-07, 3.17e-07, 3.17e-07,
00592    3.17e-07, 3.17e-07, 3.17e-07, 3.124e-07, 3.077e-07, 3.03e-07,
00593    2.984e-07, 2.938e-07, 2.892e-07, 2.847e-07, 2.779e-07, 2.705e-07,
00594    2.631e-07, 2.557e-07, 2.484e-07, 2.345e-07, 2.201e-07, 2.01e-07,
00595    1.754e-07, 1.532e-07, 1.329e-07, 1.154e-07, 1.003e-07, 8.735e-08,
00596    7.617e-08, 6.512e-08, 5.547e-08, 4.709e-08, 3.915e-08, 3.259e-08,
00597    2.738e-08, 2.327e-08, 1.98e-08, 1.711e-08, 1.493e-08, 1.306e-08,
00598    1.165e-08, 1.049e-08, 9.439e-09, 8.375e-09, 7.391e-09, 6.525e-09,
00599    5.759e-09, 5.083e-09, 4.485e-09, 3.953e-09, 3.601e-09, 3.27e-09,
00600    2.975e-09, 2.757e-09, 2.556e-09, 2.37e-09, 2.195e-09, 2.032e-09,
00601    1.912e-09, 1.79e-09, 1.679e-09, 1.572e-09, 1.482e-09, 1.402e-09,
00602    1.326e-09, 1.254e-09, 1.187e-09, 1.127e-09, 1.071e-09, 1.02e-09,
00603    9.673e-10, 9.193e-10, 8.752e-10, 8.379e-10, 8.017e-10, 7.66e-10,
00604    7.319e-10, 7.004e-10, 6.721e-10, 6.459e-10, 6.199e-10, 5.942e-10,
00605    5.703e-10, 5.488e-10, 5.283e-10, 5.082e-10, 4.877e-10, 4.696e-10,
00606    4.52e-10, 4.355e-10, 4.198e-10, 4.039e-10, 3.888e-10, 3.754e-10,
00607    3.624e-10, 3.499e-10, 3.381e-10, 3.267e-10, 3.163e-10, 3.058e-10,
00608    2.959e-10, 2.864e-10, 2.77e-10, 2.686e-10, 2.604e-10, 2.534e-10,
00609    2.462e-10, 2.386e-10, 2.318e-10, 2.247e-10, 2.189e-10, 2.133e-10,
00610    2.071e-10, 2.014e-10, 1.955e-10, 1.908e-10, 1.86e-10, 1.817e-10
00611 };
00612
```

```
00613 static double n2o5[121] = {
00614     1.231e-11, 3.035e-12, 1.702e-12, 9.877e-13, 8.081e-13, 9.039e-13,
00615     1.169e-12, 1.474e-12, 1.651e-12, 1.795e-12, 1.998e-12, 2.543e-12,
00616     4.398e-12, 7.698e-12, 1.28e-11, 2.131e-11, 3.548e-11, 5.894e-11,
00617     7.645e-11, 1.089e-10, 1.391e-10, 1.886e-10, 2.386e-10, 2.986e-10,
00618     3.487e-10, 3.994e-10, 4.5e-10, 4.6e-10, 4.591e-10, 4.1e-10, 3.488e-10,
00619     2.846e-10, 2.287e-10, 1.696e-10, 1.011e-10, 6.428e-11, 4.324e-11,
00620     2.225e-11, 6.214e-12, 3.608e-12, 8.793e-13, 4.491e-13, 1.04e-13,
00621     6.1e-14, 3.436e-14, 6.671e-15, 1.171e-15, 5.848e-16, 1.212e-16,
00622     1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00623     1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00624     1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00625     1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00626     1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00627     1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00628     1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00629     1e-16, 1e-16
00630 };
00631
00632 static double nh3[121] = {
00633     1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10,
00634     1e-10, 1e-10, 1e-10, 1e-10, 9.444e-11, 8.488e-11, 7.241e-11, 5.785e-11,
00635     4.178e-11, 3.018e-11, 2.18e-11, 1.574e-11, 1.137e-11, 8.211e-12,
00636     5.973e-12, 4.327e-12, 3.118e-12, 2.234e-12, 1.573e-12, 1.04e-12,
00637     6.762e-13, 4.202e-13, 2.406e-13, 1.335e-13, 6.938e-14, 3.105e-14,
00638     1.609e-14, 1.033e-14, 6.432e-15, 4.031e-15, 2.555e-15, 1.656e-15,
00639     1.115e-15, 7.904e-16, 5.63e-16, 4.048e-16, 2.876e-16, 2.004e-16,
00640     1.356e-16, 9.237e-17, 6.235e-17, 4.223e-17, 3.009e-17, 2.328e-17,
00641     2.002e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00642     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00643     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00644     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00645     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00646     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00647     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00648     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00649     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00650     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00651     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00652     1.914e-17
00653 };
00654
00655 static double no[121] = {
00656     2.586e-10, 4.143e-11, 1.566e-11, 9.591e-12, 8.088e-12, 8.462e-12,
00657     1.013e-11, 1.328e-11, 1.855e-11, 2.678e-11, 3.926e-11, 5.464e-11,
00658     7.012e-11, 8.912e-11, 1.127e-10, 1.347e-10, 1.498e-10, 1.544e-10,
00659     1.602e-10, 1.824e-10, 2.078e-10, 2.366e-10, 2.691e-10, 5.141e-10,
00660     8.259e-10, 1.254e-09, 1.849e-09, 2.473e-09, 3.294e-09, 4.16e-09,
00661     5.095e-09, 6.11e-09, 6.93e-09, 7.888e-09, 8.903e-09, 9.713e-09,
00662     1.052e-08, 1.115e-08, 1.173e-08, 1.21e-08, 1.228e-08, 1.239e-08,
00663     1.231e-08, 1.213e-08, 1.192e-08, 1.138e-08, 1.085e-08, 1.008e-08,
00664     9.224e-09, 8.389e-09, 7.262e-09, 6.278e-09, 5.335e-09, 4.388e-09,
00665     3.589e-09, 2.761e-09, 2.129e-09, 1.633e-09, 1.243e-09, 9.681e-10,
00666     8.355e-10, 7.665e-10, 7.442e-10, 8.584e-10, 9.732e-10, 1.063e-09,
00667     1.163e-09, 1.286e-09, 1.472e-09, 1.707e-09, 2.032e-09, 2.474e-09,
00668     2.977e-09, 3.506e-09, 4.102e-09, 5.013e-09, 6.493e-09, 8.414e-09,
00669     1.077e-08, 1.376e-08, 1.777e-08, 2.625e-08, 3.926e-08, 5.545e-08,
00670     7.195e-08, 9.464e-08, 1.404e-07, 2.183e-07, 3.329e-07, 4.535e-07,
00671     6.158e-07, 8.187e-07, 1.075e-06, 1.422e-06, 1.979e-06, 2.71e-06,
00672     3.58e-06, 4.573e-06, 5.951e-06, 7.999e-06, 1.072e-05, 1.372e-05,
00673     1.697e-05, 2.112e-05, 2.643e-05, 3.288e-05, 3.994e-05, 4.794e-05,
00674     5.606e-05, 6.383e-05, 7.286e-05, 8.156e-05, 8.883e-05, 9.469e-05,
00675     9.848e-05, 0.0001023, 0.0001066, 0.0001115, 0.0001145, 0.0001142,
00676     0.0001133
00677 };
00678
00679 static double no2[121] = {
00680     3.036e-09, 2.945e-10, 9.982e-11, 5.069e-11, 3.485e-11, 2.982e-11,
00681     2.947e-11, 3.164e-11, 3.714e-11, 4.586e-11, 6.164e-11, 8.041e-11,
00682     9.982e-11, 1.283e-10, 1.73e-10, 2.56e-10, 3.909e-10, 5.959e-10,
00683     9.081e-10, 1.384e-09, 1.788e-09, 2.189e-09, 2.686e-09, 3.091e-09,
00684     3.49e-09, 3.796e-09, 4.2e-09, 5.103e-09, 6.005e-09, 6.3e-09, 6.706e-09,
00685     7.07e-09, 7.434e-09, 7.663e-09, 7.788e-09, 7.8e-09, 7.597e-09,
00686     7.482e-09, 7.227e-09, 6.403e-09, 5.585e-09, 4.606e-09, 3.703e-09,
00687     2.984e-09, 2.183e-09, 1.48e-09, 8.441e-10, 5.994e-10, 3.799e-10,
00688     2.751e-10, 1.927e-10, 1.507e-10, 1.102e-10, 6.971e-11, 5.839e-11,
00689     3.904e-11, 3.087e-11, 2.176e-11, 1.464e-11, 1.209e-11, 8.497e-12,
00690     6.477e-12, 4.371e-12, 2.914e-12, 2.424e-12, 1.753e-12, 1.35e-12,
00691     9.417e-13, 6.622e-13, 5.148e-13, 3.841e-13, 3.446e-13, 3.01e-13,
00692     2.551e-13, 2.151e-13, 1.829e-13, 1.64e-13, 1.475e-13, 1.352e-13,
00693     1.155e-13, 9.963e-14, 9.771e-14, 9.577e-14, 9.384e-14, 9.186e-14,
00694     9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14,
00695     9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14,
00696     9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14,
00697     9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14
00698 };
00699
```



```
00700 static double o3[121] = {
00701     2.218e-08, 3.394e-08, 3.869e-08, 4.219e-08, 4.501e-08, 4.778e-08,
00702     5.067e-08, 5.402e-08, 5.872e-08, 6.521e-08, 7.709e-08, 9.461e-08,
00703     1.269e-07, 1.853e-07, 2.723e-07, 3.964e-07, 5.773e-07, 8.2e-07,
00704     1.155e-06, 1.59e-06, 2.076e-06, 2.706e-06, 3.249e-06, 3.848e-06,
00705     4.459e-06, 4.986e-06, 5.573e-06, 5.958e-06, 6.328e-06, 6.661e-06,
00706     6.9e-06, 7.146e-06, 7.276e-06, 7.374e-06, 7.447e-06, 7.383e-06,
00707     7.321e-06, 7.161e-06, 6.879e-06, 6.611e-06, 6.216e-06, 5.765e-06,
00708     5.355e-06, 4.905e-06, 4.471e-06, 4.075e-06, 3.728e-06, 3.413e-06,
00709     3.125e-06, 2.856e-06, 2.607e-06, 2.379e-06, 2.17e-06, 1.978e-06,
00710     1.8e-06, 1.646e-06, 1.506e-06, 1.376e-06, 1.233e-06, 1.102e-06,
00711     9.839e-07, 8.771e-07, 7.814e-07, 6.947e-07, 6.102e-07, 5.228e-07,
00712     4.509e-07, 3.922e-07, 3.501e-07, 3.183e-07, 2.909e-07, 2.686e-07,
00713     2.476e-07, 2.284e-07, 2.109e-07, 2.003e-07, 2.013e-07, 2.022e-07,
00714     2.032e-07, 2.042e-07, 2.097e-07, 2.361e-07, 2.656e-07, 2.989e-07,
00715     3.37e-07, 3.826e-07, 4.489e-07, 5.26e-07, 6.189e-07, 7.312e-07,
00716     8.496e-07, 8.444e-07, 8.392e-07, 8.339e-07, 8.286e-07, 8.234e-07,
00717     8.181e-07, 8.129e-07, 8.077e-07, 8.026e-07, 6.918e-07, 5.176e-07,
00718     3.865e-07, 2.885e-07, 2.156e-07, 1.619e-07, 1.219e-07, 9.161e-08,
00719     6.972e-08, 5.399e-08, 3.498e-08, 2.111e-08, 1.322e-08, 8.482e-09,
00720     5.527e-09, 3.423e-09, 2.071e-09, 1.314e-09, 8.529e-10, 5.503e-10,
00721     3.665e-10
00722 };
00723
00724 static double ocs[121] = {
00725     6e-10, 6e-10, 6e-10, 6e-10, 6e-10, 6e-10, 6e-10, 6e-10, 5.997e-10,
00726     5.989e-10, 5.881e-10, 5.765e-10, 5.433e-10, 5.074e-10, 4.567e-10,
00727     4.067e-10, 3.601e-10, 3.093e-10, 2.619e-10, 2.232e-10, 1.805e-10,
00728     1.46e-10, 1.187e-10, 8.03e-11, 5.435e-11, 3.686e-11, 2.217e-11,
00729     1.341e-11, 8.756e-12, 4.511e-12, 2.37e-12, 1.264e-12, 8.28e-13,
00730     5.263e-13, 3.209e-13, 1.717e-13, 9.068e-14, 4.709e-14, 2.389e-14,
00731     1.236e-14, 1.127e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00732     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00733     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00734     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00735     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00736     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00737     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00738     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00739     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00740     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00741     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00742     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00743     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00744     1.091e-14, 1.091e-14, 1.091e-14
00745 };
00746
00747 static double sf6[121] = {
00748     4.103e-12, 4.103e-12, 4.103e-12, 4.103e-12, 4.103e-12, 4.103e-12,
00749     4.103e-12, 4.103e-12, 4.103e-12, 4.087e-12, 4.064e-12, 4.023e-12,
00750     3.988e-12, 3.941e-12, 3.884e-12, 3.755e-12, 3.622e-12, 3.484e-12,
00751     3.32e-12, 3.144e-12, 2.978e-12, 2.811e-12, 2.653e-12, 2.489e-12,
00752     2.332e-12, 2.199e-12, 2.089e-12, 2.013e-12, 1.953e-12, 1.898e-12,
00753     1.859e-12, 1.826e-12, 1.798e-12, 1.776e-12, 1.757e-12, 1.742e-12,
00754     1.728e-12, 1.717e-12, 1.707e-12, 1.698e-12, 1.691e-12, 1.685e-12,
00755     1.679e-12, 1.675e-12, 1.671e-12, 1.668e-12, 1.665e-12, 1.663e-12,
00756     1.661e-12, 1.659e-12, 1.658e-12, 1.657e-12, 1.656e-12, 1.655e-12,
00757     1.654e-12, 1.653e-12, 1.653e-12, 1.652e-12, 1.652e-12, 1.652e-12,
00758     1.651e-12, 1.651e-12, 1.651e-12, 1.651e-12, 1.651e-12, 1.651e-12,
00759     1.651e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00760     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00761     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00762     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00763     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00764     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00765     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00766     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12
00767 };
00768
00769 static double so2[121] = {
00770     1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10,
00771     1e-10, 1e-10, 9.867e-11, 9.537e-11, 9e-11, 8.404e-11, 7.799e-11,
00772     7.205e-11, 6.616e-11, 6.036e-11, 5.475e-11, 5.007e-11, 4.638e-11,
00773     4.346e-11, 4.055e-11, 3.763e-11, 3.471e-11, 3.186e-11, 2.905e-11,
00774     2.631e-11, 2.358e-11, 2.415e-11, 2.949e-11, 3.952e-11, 5.155e-11,
00775     6.76e-11, 8.741e-11, 1.099e-10, 1.278e-10, 1.414e-10, 1.512e-10,
00776     1.607e-10, 1.699e-10, 1.774e-10, 1.832e-10, 1.871e-10, 1.907e-10,
00777     1.943e-10, 1.974e-10, 1.993e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00778     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00779     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00780     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00781     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00782     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00783     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00784     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10
00785 };
00786
```

```

00787 static int ig_co2 = -999;
00788
00789 double *q[NG] = { NULL };
00790
00791 /* Find emitter index of CO2... */
00792 if (ig_co2 == -999)
00793     ig_co2 = find_emitter(ctl, "CO2");
00794
00795 /* Identify variable... */
00796 for (int ig = 0; ig < ctl->ng; ig++) {
00797     q[ig] = NULL;
00798     if (strcasecmp(ctl->emitter[ig], "C2H2") == 0)
00799         q[ig] = c2h2;
00800     if (strcasecmp(ctl->emitter[ig], "C2H6") == 0)
00801         q[ig] = c2h6;
00802     if (strcasecmp(ctl->emitter[ig], "CCl4") == 0)
00803         q[ig] = ccl4;
00804     if (strcasecmp(ctl->emitter[ig], "CH4") == 0)
00805         q[ig] = ch4;
00806     if (strcasecmp(ctl->emitter[ig], "ClO") == 0)
00807         q[ig] = clo;
00808     if (strcasecmp(ctl->emitter[ig], "ClONO2") == 0)
00809         q[ig] = clono2;
00810     if (strcasecmp(ctl->emitter[ig], "CO") == 0)
00811         q[ig] = co;
00812     if (strcasecmp(ctl->emitter[ig], "COF2") == 0)
00813         q[ig] = cof2;
00814     if (strcasecmp(ctl->emitter[ig], "F11") == 0)
00815         q[ig] = f11;
00816     if (strcasecmp(ctl->emitter[ig], "F12") == 0)
00817         q[ig] = f12;
00818     if (strcasecmp(ctl->emitter[ig], "F14") == 0)
00819         q[ig] = f14;
00820     if (strcasecmp(ctl->emitter[ig], "F22") == 0)
00821         q[ig] = f22;
00822     if (strcasecmp(ctl->emitter[ig], "H2O") == 0)
00823         q[ig] = h2o;
00824     if (strcasecmp(ctl->emitter[ig], "H2O2") == 0)
00825         q[ig] = h2o2;
00826     if (strcasecmp(ctl->emitter[ig], "HCN") == 0)
00827         q[ig] = hcn;
00828     if (strcasecmp(ctl->emitter[ig], "HNO3") == 0)
00829         q[ig] = hno3;
00830     if (strcasecmp(ctl->emitter[ig], "HNO4") == 0)
00831         q[ig] = hno4;
00832     if (strcasecmp(ctl->emitter[ig], "HOCl") == 0)
00833         q[ig] = hocl;
00834     if (strcasecmp(ctl->emitter[ig], "N2O") == 0)
00835         q[ig] = n2o;
00836     if (strcasecmp(ctl->emitter[ig], "N2O5") == 0)
00837         q[ig] = n2o5;
00838     if (strcasecmp(ctl->emitter[ig], "NH3") == 0)
00839         q[ig] = nh3;
00840     if (strcasecmp(ctl->emitter[ig], "NO") == 0)
00841         q[ig] = no;
00842     if (strcasecmp(ctl->emitter[ig], "NO2") == 0)
00843         q[ig] = no2;
00844     if (strcasecmp(ctl->emitter[ig], "O3") == 0)
00845         q[ig] = o3;
00846     if (strcasecmp(ctl->emitter[ig], "OCS") == 0)
00847         q[ig] = ocs;
00848     if (strcasecmp(ctl->emitter[ig], "SF6") == 0)
00849         q[ig] = sf6;
00850     if (strcasecmp(ctl->emitter[ig], "SO2") == 0)
00851         q[ig] = so2;
00852 }
00853
00854 /* Loop over atmospheric data points... */
00855 for (int ip = 0; ip < atm->np; ip++) {
00856
00857     /* Get altitude index... */
00858     int iz = locate_reg(z, 121, atm->z[ip]);
00859
00860     /* Interpolate pressure... */
00861     atm->p[ip] = EXP(z[iz], pre[iz], z[iz + 1], pre[iz + 1], atm->z[ip]);
00862
00863     /* Interpolate temperature... */
00864     atm->t[ip] = LIN(z[iz], tem[iz], z[iz + 1], tem[iz + 1], atm->z[ip]);
00865
00866     /* Interpolate trace gases... */
00867     for (int ig = 0; ig < ctl->ng; ig++)
00868         if (q[ig] != NULL)
00869             atm->q[ig][ip] =
00870                 LIN(z[iz], q[ig][iz], z[iz + 1], q[ig][iz + 1], atm->z[ip]);
00871     else
00872         atm->q[ig][ip] = 0;
00873 }

```

```

00874      /* Set CO2... */
00875      if (ig_co2 >= 0)
00876          atm->q[ig_co2][ip] =
00877              371.789948e-6 + 2.026214e-6 * (atm->time[ip] - 63158400.) / 31557600.;
00878
00879      /* Set extinction to zero... */
00880      for (int iw = 0; iw < ctl->nw; iw++)
00881          atm->k[iw][ip] = 0;
00882
00883      /* Set cloud layer... */
00884      atm->clz = atm->cldz = 0;
00885      for (int icl = 0; icl < ctl->ncl; icl++)
00886          atm->clk[icl] = 0;
00887
00888      /* Set surface layer... */
00889      atm->sfz = atm->sfp = atm->sft = 0;
00890      for (int isf = 0; isf < ctl->nsf; isf++)
00891          atm->sfeps[isf] = 1;
00892  }
00893 }
00894
00895 /*****
00896
00897 double ctmc02(
00898     double nu,
00899     double p,
00900     double t,
00901     double u) {
00902
00903     static double co2296[2001] = { 9.3388e-5, 9.7711e-5, 1.0224e-4, 1.0697e-4,
00904         1.1193e-4, 1.1712e-4, 1.2255e-4, 1.2824e-4, 1.3419e-4, 1.4043e-4,
00905         1.4695e-4, 1.5378e-4, 1.6094e-4, 1.6842e-4, 1.7626e-4, 1.8447e-4,
00906         1.9307e-4, 2.0207e-4, 2.1149e-4, 2.2136e-4, 2.3169e-4, 2.4251e-4,
00907         2.5384e-4, 2.657e-4, 2.7813e-4, 2.9114e-4, 3.0477e-4, 3.1904e-4,
00908         3.3399e-4, 3.4965e-4, 3.6604e-4, 3.8322e-4, 4.0121e-4, 4.2006e-4,
00909         4.398e-4, 4.6047e-4, 4.8214e-4, 5.0483e-4, 5.286e-4, 5.535e-4,
00910         5.7959e-4, 6.0693e-4, 6.3557e-4, 6.6558e-4, 6.9702e-4, 7.2996e-4,
00911         7.6449e-4, 8.0066e-4, 8.3856e-4, 8.7829e-4, 9.1991e-4, 9.6354e-4,
00912         .0010093, .0010572, .0011074, .00116, .0012152, .001273,
00913         .0013336, .0013972, .0014638, .0015336, .0016068, .0016835,
00914         .001764, .0018483, .0019367, .0020295, .0021267, .0022286,
00915         .0023355, .0024476, .0025652, .0026885, .0028178, .0029534,
00916         .0030956, .0032448, .0034012, .0035654, .0037375, .0039181,
00917         .0041076, .0043063, .0045148, .0047336, .0049632, .005204,
00918         .0054567, .0057219, .0060002, .0062923, .0065988, .0069204,
00919         .007258, .0076123, .0079842, .0083746, .0087844, .0092146,
00920         .0096663, .01014, .010638, .011161, .01171, .012286, .012891,
00921         .013527, .014194, .014895, .015631, .016404, .017217, .01807,
00922         .018966, .019908, .020897, .021936, .023028, .024176, .025382,
00923         .026649, .027981, .02938, .030851, .032397, .034023, .035732,
00924         .037528, .039416, .041402, .04349, .045685, .047994, .050422,
00925         .052975, .055661, .058486, .061458, .064584, .067873, .071334,
00926         .074975, .078807, .082839, .087082, .091549, .096249, .1012,
00927         .10641, .11189, .11767, .12375, .13015, .13689, .14399, .15147,
00928         .15935, .16765, .17639, .18561, .19531, .20554, .21632, .22769,
00929         .23967, .25229, .2656, .27964, .29443, .31004, .3265, .34386,
00930         .36218, .3815, .40188, .42339, .44609, .47004, .49533, .52202,
00931         .5502, .57995, .61137, .64455, .6796, .71663, .75574, .79707,
00932         .84075, .88691, .9357, .98728, 1.0418, 1.0995, 1.1605, 1.225,
00933         1.2932, 1.3654, 1.4418, 1.5227, 1.6083, 1.6989, 1.7948, 1.8964,
00934         2.004, 2.118, 2.2388, 2.3668, 2.5025, 2.6463, 2.7988, 2.9606,
00935         3.1321, 3.314, 3.5071, 3.712, 3.9296, 4.1605, 4.4058, 4.6663,
00936         4.9431, 5.2374, 5.5501, 5.8818, 6.2353, 6.6114, 7.0115, 7.4372,
00937         7.8905, 8.3731, 8.8871, 9.4349, 10.019, 10.641, 11.305, 12.013,
00938         12.769, 13.576, 14.437, 15.358, 16.342, 17.39, 18.513, 19.716,
00939         21.003, 22.379, 23.854, 25.436, 27.126, 28.942, 30.89, 32.973,
00940         35.219, 37.634, 40.224, 43.021, 46.037, 49.29, 52.803, 56.447,
00941         60.418, 64.792, 69.526, 74.637, 80.182, 86.193, 92.713, 99.786,
00942         107.47, 115.84, 124.94, 134.86, 145.69, 157.49, 170.3, 184.39,
00943         199.83, 216.4, 234.55, 254.72, 276.82, 299.85, 326.16, 354.99,
00944         386.51, 416.68, 449.89, 490.12, 534.35, 578.25, 632.26, 692.61,
00945         756.43, 834.75, 924.11, 1016.9, 996.96, 1102.7, 1219.2, 1351.9,
00946         1494.3, 1654.1, 1826.5, 2027.9, 2249., 2453.8, 2714.4, 2999.4,
00947         3209.5, 3509., 3840.4, 3907.5, 4190.7, 4533.5, 4648.3, 5059.1,
00948         5561.6, 6191.4, 6820.8, 7905.9, 9362.2, 2431.3, 2211.3, 2046.8,
00949         2023.8, 1985.9, 1905.9, 1491.1, 1369.8, 1262.2, 1200.7, 887.74,
00950         820.25, 885.23, 887.21, 816.73, 1126.9, 1216.2, 1272.4, 1579.5,
00951         1634.2, 1656.3, 1657.9, 1789.5, 1670.8, 1509.5, 8474.6, 7489.2,
00952         6793.6, 6117., 5574.1, 5141.2, 5084.6, 4745.1, 4413.2, 4102.8,
00953         4024.7, 3715., 3398.6, 3100.8, 2900.4, 2629.2, 2374., 2144.7,
00954         1955.8, 1760.8, 1591.2, 1435.2, 1296.2, 1174., 1065.1, 967.76,
00955         999.48, 897.45, 809.23, 732.77, 670.26, 611.93, 560.11, 518.77,
00956         476.84, 438.8, 408.48, 380.21, 349.24, 322.71, 296.65, 272.85,
00957         251.96, 232.04, 213.88, 197.69, 182.41, 168.41, 155.79, 144.05,
00958         133.31, 123.48, 114.5, 106.21, 98.591, 91.612, 85.156, 79.204,
00959         73.719, 68.666, 63.975, 59.637, 56.35, 52.545, 49.042, 45.788,
00960         42.78, 39.992, 37.441, 35.037, 32.8, 30.744, 28.801, 26.986,

```

```

00961 25.297, 23.731, 22.258, 20.883, 19.603, 18.403, 17.295, 16.249,
00962 15.271, 14.356, 13.501, 12.701, 11.954, 11.254, 10.6, 9.9864,
00963 9.4118, 8.8745, 8.3714, 7.8997, 7.4578, 7.0446, 6.6573, 6.2949,
00964 5.9577, 5.6395, 5.3419, 5.063, 4.8037, 4.5608, 4.3452, 4.1364,
00965 3.9413, 3.7394, 3.562, 3.3932, 3.2325, 3.0789, 2.9318, 2.7898,
00966 2.6537, 2.5225, 2.3958, 2.2305, 2.1215, 2.0245, 1.9427, 1.8795,
00967 1.8336, 1.7604, 1.7016, 1.6419, 1.5282, 1.4611, 1.3443, 1.27,
00968 1.1675, 1.0824, 1.0534, .99833, .95854, .92981, .90887, .89346,
00969 .88113, .87068, .86102, .85096, .88262, .86151, .83565, .80518,
00970 .77045, .73736, .74744, .74954, .75773, .82267, .83493, .89402,
00971 .89725, .93426, .95564, .94045, .94174, .93404, .92035, .90456,
00972 .88621, .86673, .78117, .7515, .72056, .68822, .65658, .62764,
00973 .55984, .55598, .57407, .60963, .63763, .66198, .61132, .60972,
00974 .52496, .50649, .41872, .3964, .32422, .27276, .24048, .23772,
00975 .2286, .22711, .23999, .32038, .34371, .36621, .38561, .39953,
00976 .40636, .44913, .42716, .3919, .35477, .33935, .3351, .39746,
00977 .40993, .49398, .49956, .56157, .54742, .57295, .57386, .55417,
00978 .50745, .471, .43446, .39102, .34993, .31269, .27888, .24912,
00979 .22291, .19994, .17972, .16197, .14633, .13252, .12029, .10942,
00980 .099745, .091118, .083404, .076494, .070292, .064716, .059697,
00981 .055173, .051093, .047411, .044089, .041092, .038392, .035965,
00982 .033789, .031846, .030122, .028607, .02729, .026169, .025209,
00983 .024405, .023766, .023288, .022925, .022716, .022681, .022685,
00984 .022768, .023133, .023325, .023486, .024004, .024126, .024083,
00985 .023785, .024023, .023029, .021649, .021108, .019454, .017809,
00986 .017292, .016635, .017037, .018068, .018977, .018756, .017847,
00987 .016557, .016142, .014459, .012869, .012381, .010875, .0098701,
00988 .009285, .0091698, .0091701, .0096145, .010553, .01106, .012613,
00989 .014362, .015017, .016507, .017741, .01768, .017784, .0171,
00990 .016357, .016172, .017257, .018978, .020935, .021741, .023567,
00991 .025183, .025589, .026732, .027648, .028278, .028215, .02856,
00992 .029015, .029062, .028851, .028497, .027825, .027801, .026523,
00993 .02487, .022967, .022168, .020194, .018605, .017903, .018439,
00994 .019697, .020311, .020855, .020057, .018608, .016738, .015963,
00995 .013844, .011801, .011134, .0097573, .0086007, .0086226,
00996 .0083721, .0090978, .0097616, .0098426, .011317, .012853, .01447,
00997 .014657, .015771, .016351, .016079, .014829, .013431, .013185,
00998 .013207, .01448, .016176, .017971, .018265, .019526, .020455,
00999 .019797, .019802, .0194, .018176, .017505, .016197, .015339,
01000 .014401, .013213, .012203, .011186, .010236, .0093288, .0084854,
01001 .0076837, .0069375, .0062614, .0056628, .0051153, .0046015,
01002 .0041501, .003752, .0033996, .0030865, .0028077, .0025586,
01003 .0023355, .0021353, .0019553, .0017931, .0016466, .0015141,
01004 .0013941, .0012852, .0011862, .0010962, .0010142, 9.3935e-4,
01005 8.71e-4, 8.0851e-4, 7.5132e-4, 6.9894e-4, 6.5093e-4, 6.0689e-4,
01006 5.6647e-4, 5.2935e-4, 4.9525e-4, 4.6391e-4, 4.3509e-4, 4.086e-4,
01007 3.8424e-4, 3.6185e-4, 3.4126e-4, 3.2235e-4, 3.0498e-4, 2.8904e-4,
01008 2.7444e-4, 2.6106e-4, 2.4883e-4, 2.3766e-4, 2.275e-4, 2.1827e-4,
01009 2.0992e-4, 2.0239e-4, 1.9563e-4, 1.896e-4, 1.8427e-4, 1.796e-4,
01010 1.7555e-4, 1.7209e-4, 1.692e-4, 1.6687e-4, 1.6505e-4, 1.6375e-4,
01011 1.6294e-4, 1.6261e-4, 1.6274e-4, 1.6334e-4, 1.6438e-4, 1.6587e-4,
01012 1.678e-4, 1.7017e-4, 1.7297e-4, 1.762e-4, 1.7988e-4, 1.8399e-4,
01013 1.8855e-4, 1.9355e-4, 1.9902e-4, 2.0494e-4, 2.1134e-4, 2.1823e-4,
01014 2.2561e-4, 2.335e-4, 2.4192e-4, 2.5088e-4, 2.604e-4, 2.705e-4,
01015 2.8119e-4, 2.9251e-4, 3.0447e-4, 3.171e-4, 3.3042e-4, 3.4447e-4,
01016 3.5927e-4, 3.7486e-4, 3.9127e-4, 4.0854e-4, 4.267e-4, 4.4579e-4,
01017 4.6586e-4, 4.8696e-4, 5.0912e-4, 5.324e-4, 5.5685e-4, 5.8253e-4,
01018 6.0949e-4, 6.378e-4, 6.6753e-4, 6.9873e-4, 7.3149e-4, 7.6588e-4,
01019 8.0198e-4, 8.3987e-4, 8.7964e-4, 9.2139e-4, 9.6522e-4, .0010112,
01020 .0010595, .0011102, .0011634, .0012193, .001278, .0013396,
01021 .0014043, .0014722, .0015436, .0016185, .0016972, .0017799,
01022 .0018668, .001958, .0020539, .0021547, .0022606, .0023719,
01023 .002489, .002612, .0027414, .0028775, .0030206, .0031712,
01024 .0033295, .0034962, .0036716, .0038563, .0040506, .0042553,
01025 .0044709, .004698, .0049373, .0051894, .0054552, .0057354,
01026 .006031, .0063427, .0066717, .0070188, .0073854, .0077726,
01027 .0081816, .0086138, .0090709, .0095543, .010066, .010607,
01028 .011181, .011789, .012433, .013116, .013842, .014613, .015432,
01029 .016304, .017233, .018224, .019281, .020394, .021574, .022836,
01030 .024181, .025594, .027088, .028707, .030401, .032245, .034219,
01031 .036262, .038539, .040987, .043578, .04641, .04949, .052726,
01032 .056326, .0602, .064093, .068521, .073278, .077734, .083064,
01033 .088731, .093885, .1003, .1072, .11365, .12187, .13078, .13989,
01034 .15095, .16299, .17634, .19116, .20628, .22419, .24386, .26587,
01035 .28811, .31399, .34321, .36606, .39675, .42742, .44243, .47197,
01036 .49993, .49027, .51147, .52803, .48931, .49729, .5026, .43854,
01037 .441, .44766, .43414, .46151, .50029, .55247, .43855, .32115,
01038 .32607, .3431, .36119, .38029, .41179, .43996, .47144, .51853,
01039 .55362, .59122, .66338, .69877, .74001, .82923, .86907, .90361,
01040 1.0025, 1.031, 1.0559, 1.104, 1.1178, 1.1341, 1.1547, 1.351,
01041 1.4772, 1.4812, 1.4907, 1.512, 1.5442, 1.5853, 1.6358, 1.6963,
01042 1.7674, 1.8474, 1.9353, 2.0335, 2.143, 2.2592, 2.3853, 2.5217,
01043 2.6686, 2.8273, 2.9998, 3.183, 3.3868, 3.6109, 3.8564, 4.1159,
01044 4.4079, 4.7278, 5.0497, 5.3695, 5.758, 6.0834, 6.4976, 6.9312,
01045 7.38, 7.5746, 7.9833, 8.3791, 8.3956, 8.7501, 9.1067, 9.072,
01046 9.4649, 9.9112, 10.402, 10.829, 11.605, 12.54, 12.713, 10.443,
01047 10.825, 11.375, 11.955, 12.623, 13.326, 14.101, 15.041, 15.547,

```

01048 16.461, 17.439, 18.716, 19.84, 21.036, 22.642, 23.901, 25.244,  
01049 27.03, 28.411, 29.871, 31.403, 33.147, 34.744, 36.456, 39.239,  
01050 43.605, 45.162, 47.004, 49.093, 51.391, 53.946, 56.673, 59.629,  
01051 63.167, 66.576, 70.254, 74.222, 78.477, 83.034, 87.914, 93.18,  
01052 98.77, 104.74, 111.15, 117.95, 125.23, 133.01, 141.33, 150.21,  
01053 159.71, 169.89, 180.93, 192.54, 204.99, 218.34, 232.65, 248.,  
01054 264.47, 282.14, 301.13, 321.53, 343.48, 367.08, 392.5, 419.88,  
01055 449.4, 481.26, 515.64, 552.79, 592.99, 636.48, 683.61, 734.65,  
01056 789.99, 850.02, 915.14, 985.81, 1062.5, 1147.1, 1237.8, 1336.4,  
01057 1443.2, 1558.9, 1684.2, 1819.2, 1965.2, 2122.6, 2291.7, 2470.8,  
01058 2665.7, 2874.9, 3099.4, 3337.9, 3541., 3813.3, 4111.9, 4439.3,  
01059 4798.9, 5196., 5639.2, 6087.5, 6657.7, 7306.7, 8040.7, 8845.5,  
01060 9702.2, 10670., 11739., 12842., 14141., 15498., 17068., 18729.,  
01061 20557., 22559., 25248., 27664., 30207., 32915., 35611., 38081.,  
01062 40715., 43191., 41651., 42750., 43785., 44353., 44366., 44189.,  
01063 43618., 42862., 41878., 35133., 35215., 36383., 39420., 44055.,  
01064 44155., 45850., 46853., 39197., 38274., 29942., 28553., 21792.,  
01065 21228., 17106., 14955., 18181., 19557., 21427., 23728., 26301.,  
01066 28584., 30775., 32536., 33867., 40089., 39204., 37329., 34452.,  
01067 31373., 33921., 34800., 36043., 44415., 45162., 52181., 50895.,  
01068 54140., 50840., 50468., 48302., 44915., 40910., 36754., 32755.,  
01069 29093., 25860., 22962., 20448., 18247., 16326., 14645., 13165.,  
01070 11861., 10708., 9686.9, 8779.7, 7971.9, 7250.8, 6605.7, 6027.2,  
01071 5507.3, 5039.1, 4616.6, 4234.8, 3889., 3575.4, 3290.5, 3031.3,  
01072 2795.2, 2579.9, 2383.1, 2203.3, 2038.6, 1887.6, 1749.1, 1621.9,  
01073 1505., 1397.4, 1298.3, 1207., 1122.8, 1045., 973.1, 906.64,  
01074 845.16, 788.22, 735.48, 686.57, 641.21, 599.1, 559.99, 523.64,  
01075 489.85, 458.42, 429.16, 401.92, 376.54, 352.88, 330.82, 310.24,  
01076 291.03, 273.09, 256.34, 240.69, 226.05, 212.37, 199.57, 187.59,  
01077 176.37, 165.87, 156.03, 146.82, 138.17, 130.07, 122.47, 115.34,  
01078 108.65, 102.37, 96.473, 90.934, 85.73, 80.84, 76.243, 71.922,  
01079 67.858, 64.034, 60.438, 57.052, 53.866, 50.866, 48.04, 45.379,  
01080 42.872, 40.51, 38.285, 36.188, 34.211, 32.347, 30.588, 28.929,  
01081 27.362, 25.884, 24.489, 23.171, 21.929, 20.755, 19.646, 18.599,  
01082 17.61, 16.677, 15.795, 14.961, 14.174, 13.43, 12.725, 12.06,  
01083 11.431, 10.834, 10.27, 9.7361, 9.2302, 8.7518, 8.2997, 7.8724,  
01084 7.4674, 7.0848, 6.7226, 6.3794, 6.054, 5.745, 5.4525, 5.1752,  
01085 4.9121, 4.6625, 4.4259, 4.2015, 3.9888, 3.7872, 3.5961, 3.4149,  
01086 3.2431, 3.0802, 2.9257, 2.7792, 2.6402, 2.5084, 2.3834, 2.2648,  
01087 2.1522, 2.0455, 1.9441, 1.848, 1.7567, 1.6701, 1.5878, 1.5097,  
01088 1.4356, 1.3651, 1.2981, 1.2345, 1.174, 1.1167, 1.062, 1.0101,  
01089 .96087, .91414, .86986, .82781, .78777, .74971, .71339, .67882,  
01090 .64604, .61473, .58507, .55676, .52987, .5044, .48014, .45715,  
01091 .43527, .41453, .3948, .37609, .35831, .34142, .32524, .30995,  
01092 .29536, .28142, .26807, .25527, .24311, .23166, .22077, .21053,  
01093 .20081, .19143, .18261, .17407, .16603, .15833, .15089, .14385,  
01094 .13707, .13065, .12449, .11865, .11306, .10774, .10266, .097818,  
01095 .093203, .088815, .084641, .080671, .076892, .073296, .069873,  
01096 .066613, .06351, .060555, .05774, .055058, .052504, .050071,  
01097 .047752, .045543, .043438, .041432, .039521, .037699, .035962,  
01098 .034307, .032729, .031225, .029791, .028423, .02712, .025877,  
01099 .024692, .023563, .022485, .021458, .020478, .019543, .018652,  
01100 .017802, .016992, .016219, .015481, .014778, .014107, .013467,  
01101 .012856, .012274, .011718, .011188, .010682, .0102, .0097393,  
01102 .0093001, .008881, .0084812, .0080997, .0077358, .0073885,  
01103 .0070571, .0067409, .0064393, .0061514, .0058768, .0056147,  
01104 .0053647, .0051262, .0048987, .0046816, .0044745, .0042769,  
01105 .0040884, .0039088, .0037373, .0035739, .003418, .0032693,  
01106 .0031277, .0029926, .0028639, .0027413, .0026245, .0025133,  
01107 .0024074, .0023066, .0022108, .0021196, .002033, .0019507,  
01108 .0018726, .0017985, .0017282, .0016617, .0015988, .0015394,  
01109 .0014834, .0014306, .0013811, .0013346, .0012911, .0012506,  
01110 .0012131, .0011784, .0011465, .0011175, .0010912, .0010678,  
01111 .0010472, .0010295, .0010147, .001003, 9.9428e-4, 9.8883e-4,  
01112 9.8673e-4, 9.8821e-4, 9.9343e-4, .0010027, .0010164, .0010348,  
01113 .0010586, .0010882, .0011245, .0011685, .0012145, .0012666,  
01114 .0013095, .0013688, .0014048, .0014663, .0015309, .0015499,  
01115 .0016144, .0016312, .001705, .0017892, .0018499, .0019715,  
01116 .0021102, .0022442, .0024284, .0025893, .0027703, .0029445,  
01117 .0031193, .003346, .0034552, .0036906, .0037584, .0040084,  
01118 .0041934, .0044587, .0047093, .0049759, .0053421, .0055134,  
01119 .0059048, .0058663, .0061036, .0063259, .0059657, .0060653,  
01120 .0060972, .0055539, .0055653, .0055772, .005331, .0054953,  
01121 .0055919, .0058684, .006183, .0066675, .0069808, .0075142,  
01122 .0078536, .0084282, .0089454, .0094625, .0093703, .0095857,  
01123 .0099283, .010063, .010521, .0097778, .0098175, .010379, .010447,  
01124 .0105, .010617, .010706, .01078, .011177, .011212, .011304,  
01125 .011446, .011603, .011816, .012165, .012545, .013069, .013539,  
01126 .01411, .014776, .016103, .017016, .017994, .018978, .01998,  
01127 .021799, .022745, .023681, .024627, .025562, .026992, .027958,  
01128 .029013, .030154, .031402, .03228, .033651, .035272, .037088,  
01129 .039021, .041213, .043597, .045977, .04877, .051809, .054943,  
01130 .058064, .061528, .06537, .069309, .071928, .075752, .079589,  
01131 .083352, .084096, .087497, .090817, .091198, .094966, .099045,  
01132 .10429, .10867, .11518, .12269, .13126, .14087, .15161, .16388,  
01133 .16423, .1759, .18721, .19994, .21275, .22513, .23041, .24231,  
01134 .25299, .25396, .26396, .27696, .27929, .2908, .30595, .31433,

```

01135     .3282, .3429, .35944, .37467, .39277, .41245, .43326, .45649,
01136     .48152, .51897, .54686, .57877, .61263, .64962, .68983, .73945,
01137     .78619, .83537, .89622, .95002, 1.0067, 1.0742, 1.1355, 1.2007,
01138     1.2738, 1.347, 1.4254, 1.5094, 1.6009, 1.6976, 1.8019, 1.9148,
01139     2.0357, 2.166, 2.3066, 2.4579, 2.6208, 2.7966, 2.986, 3.188,
01140     3.4081, 3.6456, 3.9, 4.1747, 4.4712, 4.7931, 5.1359, 5.5097,
01141     5.9117, 6.3435, 6.8003, 7.3001, 7.8385, 8.3945, 9.011, 9.6869,
01142     10.392, 11.18, 12.036, 12.938, 13.944, 14.881, 16.029, 17.255,
01143     18.574, 19.945, 21.38, 22.9, 24.477, 26.128, 27.87, 29.037,
01144     30.988, 33.145, 35.506, 37.76, 40.885, 44.487, 48.505, 52.911,
01145     57.56, 61.964, 67.217, 72.26, 78.343, 85.08, 91.867, 99.435,
01146     107.68, 116.97, 127.12, 138.32, 150.26, 163.04, 174.81, 189.26,
01147     205.61, 224.68, 240.98, 261.88, 285.1, 307.58, 334.35, 363.53,
01148     394.68, 427.85, 458.85, 489.25, 472.87, 486.93, 496.27, 501.52,
01149     501.57, 497.14, 488.09, 476.32, 393.76, 388.51, 393.42, 414.45,
01150     455.12, 514.62, 520.38, 547.42, 562.6, 487.47, 480.83, 391.06,
01151     376.92, 303.7, 295.91, 256.03, 236.73, 280.38, 310.71, 335.53,
01152     367.88, 401.94, 435.52, 469.13, 497.94, 588.82, 597.94, 597.2,
01153     588.28, 571.2, 555.75, 603.56, 638.15, 680.75, 801.72, 848.01,
01154     962.15, 990.06, 1068.1, 1076.2, 1115.3, 1134.2, 1136.6, 1119.1,
01155     1108.9, 1090.6, 1068.7, 1041.9, 1005.4, 967.98, 927.08, 780.1,
01156     751.41, 733.12, 742.65, 785.56, 855.16, 852.45, 878.1, 784.59,
01157     777.81, 765.13, 622.93, 498.09, 474.89, 386.9, 378.48, 336.17,
01158     322.04, 329.57, 350.5, 383.38, 420.02, 462.39, 499.71, 531.98,
01159     654.99, 653.43, 639.99, 605.16, 554.16, 504.42, 540.64, 552.33,
01160     679.46, 699.51, 713.91, 832.17, 919.91, 884.96, 907.57, 846.56,
01161     818.56, 768.93, 706.71, 642.17, 575.95, 515.38, 459.07, 409.02,
01162     364.61, 325.46, 291.1, 260.89, 234.39, 211.01, 190.38, 172.11,
01163     155.91, 141.49, 128.63, 117.13, 106.84, 97.584, 89.262, 81.756,
01164     74.975, 68.842, 63.28, 58.232, 53.641, 49.46, 45.649, 42.168,
01165     38.991, 36.078, 33.409, 30.96, 28.71, 26.642, 24.737, 22.985,
01166     21.37, 19.882, 18.512, 17.242, 16.073, 14.987, 13.984, 13.05,
01167     12.186, 11.384, 10.637, 9.9436, 9.2988, 8.6991, 8.141, 7.6215,
01168     7.1378, 6.6872, 6.2671, 5.8754, 5.51, 5.1691, 4.851, 4.5539,
01169     4.2764, 4.0169, 3.7742, 3.5472, 3.3348, 3.1359, 2.9495, 2.7749,
01170     2.6113, 2.4578, 2.3139, 2.1789, 2.0523, 1.9334, 1.8219, 1.7171,
01171     1.6188, 1.5263, 1.4395, 1.3579, 1.2812, 1.209, 1.1411, 1.0773,
01172     1.0171, .96048, .90713, .85684, .80959, .76495, .72282, .68309,
01173     .64563, .61035, .57707, .54573, .51622, .48834, .46199, .43709,
01174     .41359, .39129, .37034, .35064, .33198, .31442, .29784, .28218,
01175     .26732, .25337, .24017, .22774, .21601, .20479, .19426
01176 };
01177
01178 static double co2260[2001] = { 5.7971e-5, 6.0733e-5, 6.3628e-5, 6.6662e-5,
01179     6.9843e-5, 7.3176e-5, 7.6671e-5, 8.0334e-5, 8.4175e-5, 8.8201e-5,
01180     9.2421e-5, 9.6846e-5, 0.00010149, 0.00010635, 0.00011145, 0.00011679,
01181     0.00012244, 0.00012828, 0.00013444, 0.00014094, 0.00014768, 0.00015479,
01182     0.00016224, 0.00017006, 0.00017826, 0.00018685, 0.00019587, 0.00020532,
01183     0.00021524, 0.00022565, 0.00023656, 0.0002484, 0.00026001, 0.00027261,
01184     0.00028582, 0.00029968, 0.00031422, 0.00032948, 0.00034548, 0.00036228,
01185     0.00037994, 0.00039838, 0.00041778, 0.00043814, 0.00045954, 0.00048191,
01186     0.00050543, 0.00053012, 0.00055603, 0.00058321, 0.00061175, 0.00064174,
01187     0.00067314, 0.00070614, 0.00074078, 0.00077714, 0.00081531, 0.00085538,
01188     0.00089745, 0.00094162, 0.00098798, 0.0010367, 0.0010878, 0.0011415,
01189     0.0011978, 0.001257, 0.0013191, 0.0013844, 0.001453, 0.0015249,
01190     0.0016006, 0.00168, 0.0017634, 0.001851, 0.001943, 0.0020397, 0.0021412,
01191     0.0022479, 0.00236, 0.0024778, 0.0026015, 0.0027316, 0.0028682,
01192     0.0030117, 0.0031626, 0.0033211, 0.0034877, 0.0036628, 0.0038469,
01193     0.0040403, 0.0042436, 0.0044574, 0.004682, 0.0049182, 0.0051665,
01194     0.0054276, 0.0057021, 0.0059907, 0.0062942, 0.0066133, 0.0069489,
01195     0.0073018, 0.0076729, 0.0080632, 0.0084738, 0.0089056, 0.0093599,
01196     0.0098377, 0.01034, 0.010869, 0.011426, 0.012011, 0.012627, 0.013276,
01197     0.013958, 0.014676, 0.015431, 0.016226, 0.017063, 0.017944, 0.018872,
01198     0.019848, 0.020876, 0.021958, 0.023098, 0.024298, 0.025561, 0.026892,
01199     0.028293, 0.029769, 0.031323, 0.032961, 0.034686, 0.036503, 0.038418,
01200     0.040435, 0.042561, 0.044801, 0.047161, 0.049649, 0.052271, 0.055035,
01201     0.057948, 0.061019, 0.064256, 0.06767, 0.07127, 0.075066, 0.079069,
01202     0.083291, 0.087744, 0.092441, 0.097396, 0.10262, 0.10814, 0.11396,
01203     0.1201, 0.12658, 0.13342, 0.14064, 0.14826, 0.1563, 0.1648, 0.17376,
01204     0.18323, 0.19324, 0.2038, 0.21496, 0.22674, 0.23919, 0.25234, 0.26624,
01205     0.28093, 0.29646, 0.31287, 0.33021, 0.34855, 0.36794, 0.38844, 0.41012,
01206     0.43305, 0.45731, 0.48297, 0.51011, 0.53884, 0.56924, 0.60141, 0.63547,
01207     0.67152, 0.70969, 0.75012, 0.79292, 0.83826, 0.8863, 0.93718, 0.99111,
01208     1.0482, 1.1088, 1.173, 1.2411, 1.3133, 1.3898, 1.471, 1.5571,
01209     1.6485, 1.7455, 1.8485, 1.9577, 2.0737, 2.197, 2.3278, 2.4668,
01210     2.6145, 2.7715, 2.9383, 3.1156, 3.3042, 3.5047, 3.7181, 3.9451,
01211     4.1866, 4.4437, 4.7174, 5.0089, 5.3192, 5.65, 6.0025, 6.3782,
01212     6.7787, 7.206, 7.6617, 8.1479, 8.6669, 9.221, 9.8128, 10.445,
01213     11.12, 11.843, 12.615, 13.441, 14.325, 15.271, 16.283, 17.367,
01214     18.529, 19.776, 21.111, 22.544, 24.082, 25.731, 27.504, 29.409,
01215     31.452, 33.654, 36.024, 38.573, 41.323, 44.29, 47.492, 50.951,
01216     54.608, 58.588, 62.929, 67.629, 72.712, 78.226, 84.207, 90.699,
01217     97.749, 105.42, 113.77, 122.86, 132.78, 143.61, 155.44, 168.33,
01218     182.48, 198.01, 214.87, 233.39, 253.86, 276.34, 300.3, 327.28,
01219     356.89, 389.48, 422.29, 458.99, 501.39, 548.13, 595.62, 652.74,
01220     716.54, 784.57, 866.78, 960.59, 1062.8, 1072.5, 1189.5, 1319.4,
01221     1467.6, 1630.2, 1813.7, 2016.9, 2253., 2515.3, 2773.5, 3092.8,

```

01222 3444.4, 3720.4, 4104.3, 4527.5, 4645.9, 5021.7, 5462.2, 5597.,  
01223 6110.6, 6732.5, 7513.8, 8270.6, 9640.6, 11487., 2796.1, 2680.1,  
01224 2441.6, 2404.2, 2334.8, 2215.2, 1642.5, 1477.9, 1328.1, 1223.5,  
01225 843.34, 766.96, 831.65, 834.84, 774.85, 1156.3, 1275.6, 1366.1,  
01226 1795.6, 1885., 1936.5, 1953.4, 2154.4, 2002.7, 1789.8, 10381.,  
01227 9040., 8216.5, 7384.7, 6721.9, 6187.7, 6143.8, 5703.9, 5276.6,  
01228 4873.1, 4736., 4325.3, 3927., 3554.1, 3286.1, 2950.1, 2642.4,  
01229 2368.7, 2138.9, 1914., 1719.6, 1543.9, 1388.6, 1252.1, 1132.2,  
01230 1024.1, 1025.4, 920.58, 829.59, 750.54, 685.01, 624.25, 570.14,  
01231 525.81, 481.85, 441.95, 408.71, 377.23, 345.86, 318.51, 292.26,  
01232 268.34, 247.04, 227.14, 209.02, 192.69, 177.59, 163.78, 151.26,  
01233 139.73, 129.19, 119.53, 110.7, 102.57, 95.109, 88.264, 81.948,  
01234 76.13, 70.768, 65.827, 61.251, 57.022, 53.495, 49.824, 46.443,  
01235 43.307, 40.405, 37.716, 35.241, 32.923, 30.77, 28.78, 26.915,  
01236 25.177, 23.56, 22.059, 20.654, 19.345, 18.126, 16.988, 15.93,  
01237 14.939, 14.014, 13.149, 12.343, 11.589, 10.884, 10.225, 9.6093,  
01238 9.0327, 8.4934, 7.9889, 7.5166, 7.0744, 6.6604, 6.2727, 5.9098,  
01239 5.5701, 5.2529, 4.955, 4.676, 4.4148, 4.171, 3.9426, 3.7332,  
01240 3.5347, 3.3493, 3.1677, 3.0025, 2.8466, 2.6994, 2.5601, 2.4277,  
01241 2.3016, 2.1814, 2.0664, 1.9564, 1.8279, 1.7311, 1.6427, 1.5645,  
01242 1.4982, 1.443, 1.374, 1.3146, 1.2562, 1.17, 1.1105, 1.0272,  
01243 .96863, .89718, .83654, .80226, .75908, .72431, .69573, .67174,  
01244 .65126, .63315, .61693, .60182, .58715, .59554, .57649, .55526,  
01245 .53177, .50622, .48176, .4813, .47642, .47492, .50273, .50293,  
01246 .52687, .52239, .53419, .53814, .52626, .52211, .51492, .50622,  
01247 .49746, .48841, .4792, .43534, .41999, .40349, .38586, .36799,  
01248 .35108, .31089, .30803, .3171, .33599, .35041, .36149, .32924,  
01249 .32462, .27309, .25961, .20922, .19504, .15683, .13098, .11588,  
01250 .11478, .11204, .11363, .12135, .16423, .17785, .19094, .20236,  
01251 .21084, .2154, .24108, .22848, .20871, .18797, .17963, .17834,  
01252 .21552, .22284, .26945, .27052, .30108, .28977, .29772, .29224,  
01253 .27658, .24956, .22777, .20654, .18392, .16338, .1452, .12916,  
01254 .1152, .10304, .092437, .083163, .075031, .067878, .061564,  
01255 .055976, .051018, .046609, .042679, .03917, .036032, .033223,  
01256 .030706, .02845, .026428, .024617, .022998, .021554, .02027,  
01257 .019136, .018141, .017278, .016541, .015926, .015432, .015058,  
01258 .014807, .014666, .014635, .014728, .014947, .01527, .015728,  
01259 .016345, .017026, .017798, .018839, .019752, .020636, .021886,  
01260 .022695, .02327, .023478, .024292, .023544, .022222, .021932,  
01261 .020052, .018143, .017722, .017031, .017782, .01938, .020734,  
01262 .020476, .019255, .017477, .016878, .014617, .012489, .011765,  
01263 .0099077, .0086446, .0079446, .0078644, .0079763, .008671,  
01264 .01001, .0108, .012933, .015349, .016341, .018484, .020254,  
01265 .020254, .020478, .019591, .018595, .018385, .019913, .022254,  
01266 .024847, .025809, .028053, .029924, .030212, .031367, .03222,  
01267 .032739, .032537, .03286, .033344, .033507, .033499, .033339,  
01268 .032809, .033041, .031723, .029837, .027511, .026603, .024032,  
01269 .021914, .020948, .021701, .023425, .024259, .024987, .023818,  
01270 .021768, .019223, .018144, .015282, .012604, .01163, .0097907,  
01271 .008336, .0082473, .0079582, .0088077, .009779, .010129, .012145,  
01272 .014378, .016761, .01726, .018997, .019998, .019809, .01819,  
01273 .016358, .016099, .01617, .017939, .020223, .022521, .02277,  
01274 .024279, .025247, .024222, .023989, .023224, .021493, .020362,  
01275 .018596, .017309, .015975, .014466, .013171, .011921, .01078,  
01276 .0097229, .0087612, .0078729, .0070682, .0063494, .0057156,  
01277 .0051459, .0046273, .0041712, .0037686, .0034119, .003095,  
01278 .0028126, .0025603, .0023342, .0021314, .0019489, .0017845,  
01279 .001636, .0015017, .00138, .0012697, .0011694, .0010782,  
01280 9.9507e-4, 9.1931e-4, 8.5013e-4, 7.869e-4, 7.2907e-4, 6.7611e-4,  
01281 6.2758e-4, 5.8308e-4, 5.4223e-4, 5.0473e-4, 4.7027e-4, 4.3859e-4,  
01282 4.0946e-4, 3.8265e-4, 3.5798e-4, 3.3526e-4, 3.1436e-4, 2.9511e-4,  
01283 2.7739e-4, 2.6109e-4, 2.4609e-4, 2.3229e-4, 2.1961e-4, 2.0797e-4,  
01284 1.9729e-4, 1.875e-4, 1.7855e-4, 1.7038e-4, 1.6294e-4, 1.5619e-4,  
01285 1.5007e-4, 1.4456e-4, 1.3961e-4, 1.3521e-4, 1.3131e-4, 1.2789e-4,  
01286 1.2494e-4, 1.2242e-4, 1.2032e-4, 1.1863e-4, 1.1733e-4, 1.1641e-4,  
01287 1.1585e-4, 1.1565e-4, 1.158e-4, 1.1629e-4, 1.1712e-4, 1.1827e-4,  
01288 1.1976e-4, 1.2158e-4, 1.2373e-4, 1.262e-4, 1.2901e-4, 1.3214e-4,  
01289 1.3562e-4, 1.3944e-4, 1.4361e-4, 1.4814e-4, 1.5303e-4, 1.5829e-4,  
01290 1.6394e-4, 1.6999e-4, 1.7644e-4, 1.8332e-4, 1.9063e-4, 1.984e-4,  
01291 2.0663e-4, 2.1536e-4, 2.246e-4, 2.3436e-4, 2.4468e-4, 2.5558e-4,  
01292 2.6708e-4, 2.7921e-4, 2.92e-4, 3.0548e-4, 3.1968e-4, 3.3464e-4,  
01293 3.5039e-4, 3.6698e-4, 3.8443e-4, 4.0281e-4, 4.2214e-4, 4.4248e-4,  
01294 4.6389e-4, 4.864e-4, 5.1009e-4, 5.3501e-4, 5.6123e-4, 5.888e-4,  
01295 6.1781e-4, 6.4833e-4, 6.8043e-4, 7.142e-4, 7.4973e-4, 7.8711e-4,  
01296 8.2644e-4, 8.6783e-4, 9.1137e-4, 9.5721e-4, .0010054, .0010562,  
01297 .0011096, .0011659, .0012251, .0012875, .0013532, .0014224,  
01298 .0014953, .001572, .0016529, .0017381, .0018279, .0019226,  
01299 .0020224, .0021277, .0022386, .0023557, .0024792, .0026095,  
01300 .002747, .0028921, .0030453, .0032071, .003378, .0035586,  
01301 .0037494, .003951, .0041642, .0043897, .0046282, .0048805,  
01302 .0051476, .0054304, .00573, .0060473, .0063837, .0067404,  
01303 .0071188, .0075203, .0079466, .0083994, .0088806, .0093922,  
01304 .0099366, .010516, .011134, .011792, .012494, .013244, .014046,  
01305 .014898, .015808, .016781, .017822, .018929, .020108, .02138,  
01306 .022729, .02419, .02576, .027412, .029233, .031198, .033301,  
01307 .035594, .038092, .040767, .04372, .046918, .050246, .053974,  
01308 .058009, .061976, .066586, .071537, .076209, .081856, .087998,

```

01309 .093821, .10113, .10913, .11731, .12724, .13821, .15025, .1639,
01310 .17807, .19472, .21356, .23496, .25758, .28387, .31389, .34104,
01311 .37469, .40989, .43309, .46845, .5042, .5023, .52981, .55275,
01312 .51075, .51976, .52457, .44779, .44721, .4503, .4243, .45244,
01313 .49491, .55399, .39021, .24802, .2501, .2618, .27475, .28879,
01314 .31317, .33643, .36257, .4018, .43275, .46525, .53333, .56599,
01315 .60557, .70142, .74194, .77736, .88567, .91182, .93294, .98407,
01316 .98772, .99176, .9995, 1.2405, 1.3602, 1.338, 1.3255, 1.3267,
01317 1.3404, 1.3634, 1.3967, 1.4407, 1.4961, 1.5603, 1.6328, 1.7153,
01318 1.8094, 1.9091, 2.018, 2.1367, 2.264, 2.4035, 2.5562, 2.7179,
01319 2.9017, 3.1052, 3.3304, 3.5731, 3.8488, 4.1553, 4.4769, 4.7818,
01320 5.1711, 5.5204, 5.9516, 6.4097, 6.8899, 7.1118, 7.5469, 7.9735,
01321 7.9511, 8.3014, 8.6418, 8.4757, 8.8256, 9.2294, 9.6923, 10.033,
01322 10.842, 11.851, 11.78, 8.8435, 9.1381, 9.5956, 10.076, 10.629,
01323 11.22, 11.883, 12.69, 13.163, 13.974, 14.846, 16.027, 17.053,
01324 18.148, 19.715, 20.907, 22.163, 23.956, 25.235, 26.566, 27.94,
01325 29.576, 30.956, 32.432, 35.337, 39.911, 41.128, 42.625, 44.386,
01326 46.369, 48.619, 51.031, 53.674, 56.825, 59.921, 63.286, 66.929,
01327 70.859, 75.081, 79.618, 84.513, 89.739, 95.335, 101.35, 107.76,
01328 114.63, 121.98, 129.87, 138.3, 147.34, 157.04, 167.56, 178.67,
01329 190.61, 203.43, 217.19, 231.99, 247.88, 264.98, 283.37, 303.17,
01330 324.49, 347.47, 372.25, 398.98, 427.85, 459.06, 492.8, 529.31,
01331 568.89, 611.79, 658.35, 708.91, 763.87, 823.65, 888.72, 959.58,
01332 1036.8, 1121.8, 1213.9, 1314.3, 1423.8, 1543., 1672.8, 1813.4,
01333 1966.1, 2131.4, 2309.5, 2499.3, 2705., 2925.7, 3161.6, 3411.3,
01334 3611.5, 3889.2, 4191.1, 4519.3, 4877.9, 5272.9, 5712.9, 6142.7,
01335 6719.6, 7385., 8145., 8977.7, 9831.9, 10827., 11934., 13063.,
01336 14434., 15878., 17591., 19435., 21510., 23835., 26835., 29740.,
01337 32878., 36305., 39830., 43273., 46931., 50499., 49586., 51598.,
01338 53429., 54619., 55081., 55102., 54485., 53487., 52042., 42689.,
01339 42607., 44020., 47994., 54169., 53916., 55808., 56642., 46049.,
01340 44243., 32929., 30658., 21963., 20835., 15962., 13679., 17652.,
01341 19680., 22388., 25625., 29184., 32520., 35720., 38414., 40523.,
01342 49228., 48173., 45678., 41768., 37600., 41313., 42654., 44465.,
01343 55736., 56630., 65409., 63308., 66572., 61845., 60379., 56777.,
01344 51920., 46601., 41367., 36529., 32219., 28470., 25192., 22362.,
01345 19907., 17772., 15907., 14273., 12835., 11567., 10445., 9450.2,
01346 8565.1, 7776., 7070.8, 6439.2, 5872.3, 5362.4, 4903., 4488.3,
01347 4113.4, 3773.8, 3465.8, 3186.1, 2931.7, 2700.1, 2488.8, 2296.,
01348 2119.8, 1958.6, 1810.9, 1675.6, 1551.4, 1437.3, 1332.4, 1236.,
01349 1147.2, 1065.3, 989.86, 920.22, 855.91, 796.48, 741.53, 690.69,
01350 643.62, 600.02, 559.6, 522.13, 487.35, 455.06, 425.08, 397.21,
01351 371.3, 347.2, 324.78, 303.9, 284.46, 266.34, 249.45, 233.7,
01352 219.01, 205.3, 192.5, 180.55, 169.38, 158.95, 149.2, 140.07,
01353 131.54, 123.56, 116.09, 109.09, 102.54, 96.405, 90.655, 85.266,
01354 80.213, 75.475, 71.031, 66.861, 62.948, 59.275, 55.827, 52.587,
01355 49.544, 46.686, 43.998, 41.473, 39.099, 36.867, 34.768, 32.795,
01356 30.939, 29.192, 27.546, 25.998, 24.539, 23.164, 21.869, 20.65,
01357 19.501, 18.419, 17.399, 16.438, 15.532, 14.678, 13.874, 13.115,
01358 12.4, 11.726, 11.088, 10.488, 9.921, 9.3846, 8.8784, 8.3996,
01359 7.9469, 7.5197, 7.1174, 6.738, 6.379, 6.0409, 5.7213, 5.419,
01360 5.1327, 4.8611, 4.6046, 4.3617, 4.1316, 3.9138, 3.7077, 3.5125,
01361 3.3281, 3.1536, 2.9885, 2.8323, 2.6846, 2.5447, 2.4124, 2.2871,
01362 2.1686, 2.0564, 1.9501, 1.8495, 1.7543, 1.6641, 1.5787, 1.4978,
01363 1.4212, 1.3486, 1.2799, 1.2147, 1.1529, 1.0943, 1.0388, .98602,
01364 .93596, .8886, .84352, .80078, .76029, .722, .68585, .65161,
01365 .61901, .58808, .55854, .53044, .5039, .47853, .45459, .43173,
01366 .41008, .38965, .37021, .35186, .33444, .31797, .30234, .28758,
01367 .2736, .26036, .24764, .2357, .22431, .21342, .20295, .19288,
01368 .18334, .17444, .166, .15815, .15072, .14348, .13674, .13015,
01369 .12399, .11807, .11231, .10689, .10164, .096696, .091955,
01370 .087476, .083183, .079113, .075229, .071536, .068026, .064698,
01371 .06154, .058544, .055699, .052997, .050431, .047993, .045676,
01372 .043475, .041382, .039392, .037501, .035702, .033991, .032364,
01373 .030817, .029345, .027945, .026613, .025345, .024139, .022991,
01374 .021899, .02086, .019871, .018929, .018033, .01718, .016368,
01375 .015595, .014859, .014158, .013491, .012856, .012251, .011675,
01376 .011126, .010604, .010107, .0096331, .009182, .0087523, .0083431,
01377 .0079533, .0075821, .0072284, .0068915, .0065706, .0062649,
01378 .0059737, .0056963, .005432, .0051802, .0049404, .0047118,
01379 .0044941, .0042867, .0040891, .0039009, .0037216, .0035507,
01380 .003388, .0032329, .0030852, .0029445, .0028105, .0026829,
01381 .0025613, .0024455, .0023353, .0022303, .0021304, .0020353,
01382 .0019448, .0018587, .0017767, .0016988, .0016247, .0015543,
01383 .0014874, .0014238, .0013635, .0013062, .0012519, .0012005,
01384 .0011517, .0011057, .0010621, .001021, 9.8233e-4, 9.4589e-4,
01385 9.1167e-4, 8.7961e-4, 8.4964e-4, 8.2173e-4, 7.9582e-4, 7.7189e-4,
01386 7.499e-4, 7.2983e-4, 7.1167e-4, 6.9542e-4, 6.8108e-4, 6.6866e-4,
01387 6.5819e-4, 6.4971e-4, 6.4328e-4, 6.3895e-4, 6.3681e-4, 6.3697e-4,
01388 6.3956e-4, 6.4472e-4, 6.5266e-4, 6.6359e-4, 6.778e-4, 6.9563e-4,
01389 7.1749e-4, 7.4392e-4, 7.7556e-4, 8.1028e-4, 8.4994e-4, 8.8709e-4,
01390 9.3413e-4, 9.6953e-4, .0010202, .0010738, .0010976, .0011507,
01391 .0011686, .0012264, .001291, .0013346, .0014246, .0015293,
01392 .0016359, .0017824, .0019255, .0020854, .002247, .0024148,
01393 .0026199, .0027523, .0029704, .0030702, .0033047, .0035013,
01394 .0037576, .0040275, .0043089, .0046927, .0049307, .0053486,
01395 .0053809, .0056699, .0059325, .0055488, .005634, .0056392,

```



```

01396 .004946, .0048855, .0048208, .0044386, .0045498, .0046377,
01397 .0048939, .0052396, .0057324, .0060859, .0066906, .0071148,
01398 .0077224, .0082687, .008769, .0084471, .008572, .0087729,
01399 .008775, .0090742, .0080704, .0080288, .0085747, .0086087,
01400 .0086408, .0088752, .0089381, .0089757, .0093532, .0092824,
01401 .0092566, .0092645, .0092735, .009342, .0095806, .0097991,
01402 .010213, .010611, .011129, .011756, .013237, .01412, .015034,
01403 .015936, .01682, .018597, .019315, .019995, .020658, .021289,
01404 .022363, .022996, .023716, .024512, .025434, .026067, .027118,
01405 .028396, .029865, .031442, .033253, .03525, .037296, .039701,
01406 .042356, .045154, .048059, .051294, .054893, .058636, .061407,
01407 .065172, .068974, .072676, .073379, .076547, .079556, .079134,
01408 .082308, .085739, .090192, .09359, .099599, .10669, .11496,
01409 .1244, .13512, .14752, .14494, .15647, .1668, .17863, .19029,
01410 .20124, .20254, .21179, .21982, .21625, .22364, .23405, .23382,
01411 .2434, .25708, .26406, .27621, .28909, .30395, .31717, .33271,
01412 .3496, .36765, .38774, .40949, .446, .46985, .49846, .5287, .562,
01413 .59841, .64598, .68834, .7327, .78978, .8373, .88708, .94744,
01414 1.0006, 1.0574, 1.1215, 1.1856, 1.2546, 1.3292, 1.4107, 1.4974,
01415 1.5913, 1.6931, 1.8028, 1.9212, 2.0492, 2.1874, 2.3365, 2.4978,
01416 2.6718, 2.8588, 3.062, 3.2818, 3.5188, 3.7752, 4.0527, 4.3542,
01417 4.6782, 5.0312, 5.4123, 5.8246, 6.2639, 6.7435, 7.2636, 7.8064,
01418 8.4091, 9.0696, 9.7677, 10.548, 11.4, 12.309, 13.324, 14.284,
01419 15.445, 16.687, 18.019, 19.403, 20.847, 22.366, 23.925, 25.537,
01420 27.213, 28.069, 29.864, 31.829, 33.988, 35.856, 38.829, 42.321,
01421 46.319, 50.606, 55.126, 59.126, 64.162, 68.708, 74.615, 81.176,
01422 87.739, 95.494, 103.83, 113.38, 123.99, 135.8, 148.7, 162.58,
01423 176.32, 192.6, 211.47, 232.7, 252.64, 277.41, 305.38, 333.44,
01424 366.42, 402.66, 442.14, 484.53, 526.42, 568.15, 558.78, 582.6,
01425 600.98, 613.94, 619.44, 618.24, 609.84, 595.96, 484.86, 475.59,
01426 478.49, 501.56, 552.19, 628.44, 630.39, 658.92, 671.96, 562.7,
01427 545.88, 423.43, 400.14, 306.59, 294.13, 246.8, 226.51, 278.21,
01428 314.39, 347.22, 389.13, 433.16, 477.48, 521.67, 560.54, 683.6,
01429 696.37, 695.91, 683.1, 658.24, 634.89, 698.85, 742.87, 796.66,
01430 954.49, 1009.5, 1150.5, 1179.1, 1267.9, 1272.4, 1312.7, 1330.4,
01431 1331.6, 1315.8, 1308.3, 1293.3, 1274.6, 1249.5, 1213.2, 1172.1,
01432 1124.4, 930.33, 893.36, 871.27, 883.54, 940.76, 1036., 1025.6,
01433 1053.1, 914.51, 894.15, 865.03, 670.63, 508.41, 475.15, 370.85,
01434 361.06, 319.38, 312.75, 331.87, 367.13, 415., 467.94, 525.49,
01435 578.41, 624.66, 794.82, 796.97, 780.29, 736.49, 670.18, 603.75,
01436 659.67, 679.8, 857.12, 884.05, 900.65, 1046.1, 1141.9, 1083.,
01437 1089.2, 1e3, 947.08, 872.31, 787.91, 704.75, 624.93, 553.68,
01438 489.91, 434.21, 385.64, 343.3, 306.42, 274.18, 245.94, 221.11,
01439 199.23, 179.88, 162.73, 147.48, 133.88, 121.73, 110.86, 101.1,
01440 92.323, 84.417, 77.281, 70.831, 64.991, 59.694, 54.884, 50.509,
01441 46.526, 42.893, 39.58, 36.549, 33.776, 31.236, 28.907, 26.77,
01442 24.805, 23., 21.339, 19.81, 18.404, 17.105, 15.909, 14.801,
01443 13.778, 12.83, 11.954, 11.142, 10.389, 9.691, 9.0434, 8.4423,
01444 7.8842, 7.3657, 6.8838, 6.4357, 6.0189, 5.6308, 5.2696, 4.9332,
01445 4.6198, 4.3277, 4.0553, 3.8012, 3.5639, 3.3424, 3.1355, 2.9422,
01446 2.7614, 2.5924, 2.4343, 2.2864, 2.148, 2.0184, 1.8971, 1.7835,
01447 1.677, 1.5773, 1.4838, 1.3961, 1.3139, 1.2369, 1.1645, 1.0966,
01448 1.0329, .97309, .91686, .86406, .81439, .76767, .72381, .68252,
01449 .64359, .60695, .57247, .54008, .50957, .48092, .45401, .42862,
01450 .40465, .38202, .36072, .34052, .3216, .30386, .28711, .27135,
01451 .25651, .24252, .2293, .21689, .20517, .19416, .18381, .17396,
01452 .16469
01453 };
01454
01455 static double co2230[2001] = { 2.743e-5, 2.8815e-5, 3.027e-5, 3.1798e-5,
01456 3.3405e-5, 3.5094e-5, 3.6869e-5, 3.8734e-5, 4.0694e-5, 4.2754e-5,
01457 4.492e-5, 4.7196e-5, 4.9588e-5, 5.2103e-5, 5.4747e-5, 5.7525e-5,
01458 6.0446e-5, 6.3516e-5, 6.6744e-5, 7.0137e-5, 7.3704e-5, 7.7455e-5,
01459 8.1397e-5, 8.5543e-5, 8.9901e-5, 9.4484e-5, 9.9302e-5, 1.0437e-4,
01460 1.097e-4, 1.153e-4, 1.2119e-4, 1.2738e-4, 1.3389e-4, 1.4074e-4,
01461 1.4795e-4, 1.5552e-4, 1.6349e-4, 1.7187e-4, 1.8068e-4, 1.8995e-4,
01462 1.997e-4, 2.0996e-4, 2.2075e-4, 2.321e-4, 2.4403e-4, 2.5659e-4,
01463 2.698e-4, 2.837e-4, 2.9832e-4, 3.137e-4, 3.2988e-4, 3.4691e-4,
01464 3.6483e-4, 3.8368e-4, 4.0351e-4, 4.2439e-4, 4.4635e-4, 4.6947e-4,
01465 4.9379e-4, 5.1939e-4, 5.4633e-4, 5.7468e-4, 6.0452e-4, 6.3593e-4,
01466 6.69e-4, 7.038e-4, 7.4043e-4, 7.79e-4, 8.1959e-4, 8.6233e-4,
01467 9.0732e-4, 9.5469e-4, .0010046, .0010571, .0011124, .0011706,
01468 .0012319, .0012964, .0013644, .001436, .0015114, .0015908,
01469 .0016745, .0017625, .0018553, .0019531, .002056, .0021645,
01470 .0022788, .0023992, .002526, .0026596, .0028004, .0029488,
01471 .0031052, .0032699, .0034436, .0036265, .0038194, .0040227,
01472 .0042369, .0044628, .0047008, .0049518, .0052164, .0054953,
01473 .0057894, .0060995, .0064265, .0067713, .007135, .0075184,
01474 .0079228, .0083494, .0087993, .0092738, .0097745, .010303,
01475 .01086, .011448, .012068, .012722, .013413, .014142, .014911,
01476 .015723, .01658, .017484, .018439, .019447, .020511, .021635,
01477 .022821, .024074, .025397, .026794, .02827, .029829, .031475,
01478 .033215, .035052, .036994, .039045, .041213, .043504, .045926,
01479 .048485, .05119, .05405, .057074, .060271, .063651, .067225,
01480 .071006, .075004, .079233, .083708, .088441, .093449, .098749,
01481 .10436, .11029, .11657, .12322, .13026, .13772, .14561, .15397,
01482 .16282, .1722, .18214, .19266, .20381, .21563, .22816, .24143,

```

```

01483 .2555, .27043, .28625, .30303, .32082, .3397, .35972, .38097,
01484 .40352, .42746, .45286, .47983, .50847, .53888, .57119, .6055,
01485 .64196, .6807, .72187, .76564, .81217, .86165, .91427, .97025,
01486 1.0298, 1.0932, 1.1606, 1.2324, 1.3088, 1.3902, 1.477, 1.5693,
01487 1.6678, 1.7727, 1.8845, 2.0038, 2.131, 2.2666, 2.4114, 2.5659,
01488 2.7309, 2.907, 3.0951, 3.2961, 3.5109, 3.7405, 3.986, 4.2485,
01489 4.5293, 4.8299, 5.1516, 5.4961, 5.8651, 6.2605, 6.6842, 7.1385,
01490 7.6256, 8.1481, 8.7089, 9.3109, 9.9573, 10.652, 11.398, 12.2,
01491 13.063, 13.992, 14.99, 16.064, 17.222, 18.469, 19.813, 21.263,
01492 22.828, 24.516, 26.34, 28.31, 30.437, 32.738, 35.226, 37.914,
01493 40.824, 43.974, 47.377, 51.061, 55.011, 59.299, 63.961, 69.013,
01494 74.492, 80.444, 86.919, 93.836, 101.23, 109.25, 117.98, 127.47,
01495 137.81, 149.07, 161.35, 174.75, 189.42, 205.49, 223.02, 242.26,
01496 263.45, 286.75, 311.94, 340.01, 370.86, 404.92, 440.44, 480.27,
01497 525.17, 574.71, 626.22, 686.8, 754.38, 827.07, 913.38, 1011.7,
01498 1121.5, 1161.6, 1289.5, 1432.2, 1595.4, 1777., 1983.3, 2216.1,
01499 2485.7, 2788.3, 3101.5, 3481., 3902.1, 4257.1, 4740., 5272.8,
01500 5457.9, 5946.25, 6505.3, 6668.4, 7302.4, 8061.6, 9015.8, 9908.3,
01501 11613., 13956., 3249.6, 3243., 2901.5, 2841.3, 2729.6, 2558.2,
01502 1797.8, 1583.2, 1386., 1233.5, 787.74, 701.46, 761.66, 767.21,
01503 722.83, 1180.6, 1332.1, 1461.6, 2032.9, 2166., 2255.9, 2294.7,
01504 2587.2, 2396.5, 2122.4, 12553., 10784., 9832.5, 8827.3, 8029.1,
01505 7377.9, 7347.1, 6783.8, 6239.1, 5721.1, 5503., 4975.1, 4477.8,
01506 4021.3, 3676.8, 3275.3, 2914.9, 2597.4, 2328.2, 2075.4, 1857.6,
01507 1663.6, 1493.3, 1343.8, 1213.3, 1095.6, 1066.5, 958.91, 865.15,
01508 783.31, 714.35, 650.77, 593.98, 546.2, 499.9, 457.87, 421.75,
01509 387.61, 355.25, 326.62, 299.7, 275.21, 253.17, 232.83, 214.31,
01510 197.5, 182.08, 167.98, 155.12, 143.32, 132.5, 122.58, 113.48,
01511 105.11, 97.415, 90.182, 83.463, 77.281, 71.587, 66.341, 61.493,
01512 57.014, 53.062, 49.21, 45.663, 42.38, 39.348, 36.547, 33.967,
01513 31.573, 29.357, 27.314, 25.415, 23.658, 22.03, 20.524, 19.125,
01514 17.829, 16.627, 15.511, 14.476, 13.514, 12.618, 11.786, 11.013,
01515 10.294, 9.6246, 9.0018, 8.4218, 7.8816, 7.3783, 6.9092, 6.4719,
01516 6.0641, 5.6838, 5.3289, 4.998, 4.6893, 4.4014, 4.1325, 3.8813,
01517 3.6469, 3.4283, 3.2241, 3.035, 2.8576, 2.6922, 2.5348, 2.3896,
01518 2.2535, 2.1258, 2.0059, 1.8929, 1.7862, 1.6854, 1.5898, 1.4992,
01519 1.4017, 1.3218, 1.2479, 1.1809, 1.1215, 1.0693, 1.0116, .96016,
01520 .9105, .84859, .80105, .74381, .69982, .65127, .60899, .57843,
01521 .54592, .51792, .49336, .47155, .45201, .43426, .41807, .40303,
01522 .38876, .3863, .37098, .35492, .33801, .32032, .30341, .29874,
01523 .29193, .28689, .29584, .29155, .29826, .29195, .29287, .2904,
01524 .28199, .27709, .27162, .26622, .26133, .25676, .25235, .23137,
01525 .22365, .21519, .20597, .19636, .18699, .16485, .16262, .16643,
01526 .17542, .18198, .18631, .16759, .16338, .13505, .1267, .10053,
01527 .092554, .074093, .062159, .055523, .054849, .05401, .05528,
01528 .058982, .07952, .08647, .093244, .099285, .10393, .10661,
01529 .12072, .11417, .10396, .093265, .089137, .088909, .10902,
01530 .11277, .13625, .13565, .14907, .14167, .1428, .13744, .12768,
01531 .11382, .10244, .091686, .08109, .071739, .063616, .056579,
01532 .050504, .045251, .040689, .036715, .033237, .030181, .027488,
01533 .025107, .022998, .021125, .01946, .017979, .016661, .015489,
01534 .014448, .013526, .012712, .011998, .011375, .010839, .010384,
01535 .010007, .0097053, .0094783, .0093257, .0092489, .0092504,
01536 .0093346, .0095077, .0097676, .01012, .01058, .011157, .011844,
01537 .012672, .013665, .014766, .015999, .017509, .018972, .020444,
01538 .022311, .023742, .0249, .025599, .026981, .026462, .025143,
01539 .025066, .022814, .020458, .020026, .019142, .020189, .022371,
01540 .024163, .023728, .02199, .019506, .018591, .015576, .012784,
01541 .011744, .0094777, .0079148, .0070652, .006986, .0071758,
01542 .008086, .0098025, .01087, .013609, .016764, .018137, .021061,
01543 .023498, .023576, .023965, .022828, .021519, .021283, .023364,
01544 .026457, .029782, .030856, .033486, .035515, .035543, .036558,
01545 .037198, .037472, .037045, .037284, .03777, .038085, .038366,
01546 .038526, .038282, .038915, .037697, .035667, .032941, .031959,
01547 .028692, .025918, .024596, .025592, .027873, .028935, .02984,
01548 .028148, .025305, .021912, .020454, .016732, .013357, .01205,
01549 .009731, .0079881, .0077704, .0074387, .0083895, .0096776,
01550 .010326, .01293, .015955, .019247, .020145, .02267, .024231,
01551 .024184, .022131, .019784, .01955, .01971, .022119, .025116,
01552 .027978, .028107, .029808, .030701, .029164, .028551, .027286,
01553 .024946, .023259, .020982, .019221, .017471, .015643, .014074,
01554 .01261, .011301, .010116, .0090582, .0081036, .0072542, .0065034,
01555 .0058436, .0052571, .0047321, .0042697, .0038607, .0034977,
01556 .0031747, .0028864, .0026284, .002397, .002189, .0020017,
01557 .0018326, .0016798, .0015414, .0014159, .0013019, .0011983,
01558 .0011039, .0010177, 9.391e-4, 8.6717e-4, 8.0131e-4, 7.4093e-4,
01559 6.8553e-4, 6.3464e-4, 5.8787e-4, 5.4487e-4, 5.0533e-4, 4.69e-4,
01560 4.3556e-4, 4.0474e-4, 3.7629e-4, 3.5e-4, 3.2569e-4, 3.032e-4,
01561 2.8239e-4, 2.6314e-4, 2.4535e-4, 2.2891e-4, 2.1374e-4, 1.9975e-4,
01562 1.8685e-4, 1.7498e-4, 1.6406e-4, 1.5401e-4, 1.4479e-4, 1.3633e-4,
01563 1.2858e-4, 1.2148e-4, 1.1499e-4, 1.0907e-4, 1.0369e-4, 9.8791e-5,
01564 9.4359e-5, 9.0359e-5, 8.6766e-5, 8.3555e-5, 8.0703e-5, 7.8192e-5,
01565 7.6003e-5, 7.4119e-5, 7.2528e-5, 7.1216e-5, 7.0171e-5, 6.9385e-5,
01566 6.8848e-5, 6.8554e-5, 6.8496e-5, 6.8669e-5, 6.9069e-5, 6.9694e-5,
01567 7.054e-5, 7.1608e-5, 7.2896e-5, 7.4406e-5, 7.6139e-5, 7.8097e-5,
01568 8.0283e-5, 8.2702e-5, 8.5357e-5, 8.8255e-5, 9.1402e-5, 9.4806e-5,
01569 9.8473e-5, 1.0241e-4, 1.0664e-4, 1.1115e-4, 1.1598e-4, 1.2112e-4,

```

01570 1.2659e-4, 1.3241e-4, 1.3859e-4, 1.4515e-4, 1.521e-4, 1.5947e-4,  
01571 1.6728e-4, 1.7555e-4, 1.8429e-4, 1.9355e-4, 2.0334e-4, 2.1369e-4,  
01572 2.2463e-4, 2.3619e-4, 2.4841e-4, 2.6132e-4, 2.7497e-4, 2.8938e-4,  
01573 3.0462e-4, 3.2071e-4, 3.3771e-4, 3.5567e-4, 3.7465e-4, 3.947e-4,  
01574 4.1588e-4, 4.3828e-4, 4.6194e-4, 4.8695e-4, 5.1338e-4, 5.4133e-4,  
01575 5.7087e-4, 6.0211e-4, 6.3515e-4, 6.701e-4, 7.0706e-4, 7.4617e-4,  
01576 7.8756e-4, 8.3136e-4, 8.7772e-4, 9.2681e-4, 9.788e-4, .0010339,  
01577 .0010922, .001154, .0012195, .0012889, .0013626, .0014407,  
01578 .0015235, .0016114, .0017048, .0018038, .001909, .0020207,  
01579 .0021395, .0022657, .0023998, .0025426, .0026944, .002856,  
01580 .0030281, .0032114, .0034068, .003615, .0038371, .004074,  
01581 .004327, .0045971, .0048857, .0051942, .0055239, .0058766,  
01582 .0062538, .0066573, .0070891, .007551, .0080455, .0085747,  
01583 .0091412, .0097481, .010397, .011092, .011837, .012638, .013495,  
01584 .014415, .01541, .016475, .017621, .018857, .020175, .02162,  
01585 .023185, .024876, .02672, .028732, .030916, .033319, .035939,  
01586 .038736, .041847, .04524, .048715, .052678, .056977, .061203,  
01587 .066184, .07164, .076952, .083477, .090674, .098049, .10697,  
01588 .1169, .1277, .14011, .15323, .1684, .18601, .20626, .22831,  
01589 .25417, .28407, .31405, .34957, .38823, .41923, .46026, .50409,  
01590 .51227, .54805, .57976, .53818, .55056, .557, .46741, .46403,  
01591 .4636, .42265, .45166, .49852, .56663, .34306, .17779, .17697,  
01592 .18346, .19129, .20014, .21778, .23604, .25649, .28676, .31238,  
01593 .33856, .39998, .4288, .46568, .56654, .60786, .64473, .76466,  
01594 .7897, .80778, .86443, .85736, .84798, .84157, 1.1385, 1.2446,  
01595 1.1923, 1.1552, 1.1338, 1.1266, 1.1292, 1.1431, 1.1683, 1.2059,  
01596 1.2521, 1.3069, 1.3712, 1.4471, 1.5275, 1.6165, 1.7145, 1.8189,  
01597 1.9359, 2.065, 2.2007, 2.3591, 2.5362, 2.7346, 2.9515, 3.2021,  
01598 3.4851, 3.7935, 4.0694, 4.4463, 4.807, 5.2443, 5.7178, 6.2231,  
01599 6.4796, 6.9461, 7.4099, 7.3652, 7.7182, 8.048, 7.7373, 8.0363,  
01600 8.3855, 8.8044, 9.0257, 9.8574, 10.948, 10.563, 6.8979, 7.0744,  
01601 7.4121, 7.7663, 8.1768, 8.6243, 9.1437, 9.7847, 10.182, 10.849,  
01602 11.572, 12.602, 13.482, 14.431, 15.907, 16.983, 18.11, 19.884,  
01603 21.02, 22.18, 23.355, 24.848, 25.954, 27.13, 30.186, 34.893,  
01604 35.682, 36.755, 38.111, 39.703, 41.58, 43.606, 45.868, 48.573,  
01605 51.298, 54.291, 57.559, 61.116, 64.964, 69.124, 73.628, 78.471,  
01606 83.683, 89.307, 95.341, 101.84, 108.83, 116.36, 124.46, 133.18,  
01607 142.57, 152.79, 163.69, 175.43, 188.11, 201.79, 216.55, 232.51,  
01608 249.74, 268.38, 288.54, 310.35, 333.97, 359.55, 387.26, 417.3,  
01609 449.88, 485.2, 523.54, 565.14, 610.28, 659.31, 712.56, 770.43,  
01610 833.36, 901.82, 976.36, 1057.6, 1146.8, 1243.8, 1350., 1466.3,  
01611 1593.6, 1732.7, 1884.1, 2049.1, 2228.2, 2421.9, 2629.4, 2853.7,  
01612 3094.4, 3351.1, 3622.3, 3829.8, 4123.1, 4438.3, 4777.2, 5144.1,  
01613 5545.4, 5990.5, 6404.5, 6996.8, 7687.6, 8482.9, 9349.4, 10203.,  
01614 11223., 12358., 13493., 14916., 16416., 18236., 20222., 22501.,  
01615 25102., 28358., 31707., 35404., 39538., 43911., 48391., 53193.,  
01616 58028., 58082., 61276., 64193., 66294., 67480., 67921., 67423.,  
01617 66254., 64341., 51737., 51420., 53072., 58145., 66195., 65358.,  
01618 67377., 67869., 53509., 50553., 35737., 32425., 21704., 19974.,  
01619 14457., 12142., 16798., 19489., 23049., 27270., 31910., 36457.,  
01620 40877., 44748., 47876., 59793., 58626., 55454., 50337., 44893.,  
01621 50228., 52216., 54747., 69541., 70455., 81014., 77694., 80533.,  
01622 73953., 70927., 65539., 59002., 52281., 45953., 40292., 35360.,  
01623 31124., 27478., 24346., 21647., 19308., 17271., 15491., 13927.,  
01624 12550., 11331., 10250., 9288.8, 8431.4, 7664.9, 6978.3, 6361.8,  
01625 5807.4, 5307.7, 4856.8, 4449., 4079.8, 3744.9, 3440.8, 3164.2,  
01626 2912.3, 2682.7, 2473., 2281.4, 2106., 1945.3, 1797.9, 1662.5,  
01627 1538.1, 1423.6, 1318.1, 1221., 1131.5, 1049., 972.99, 902.87,  
01628 838.01, 777.95, 722.2, 670.44, 622.35, 577.68, 536.21, 497.76,  
01629 462.12, 429.13, 398.61, 370.39, 344.29, 320.16, 297.85, 277.2,  
01630 258.08, 240.38, 223.97, 208.77, 194.66, 181.58, 169.43, 158.15,  
01631 147.67, 137.92, 128.86, 120.44, 112.6, 105.3, 98.499, 92.166,  
01632 86.264, 80.763, 75.632, 70.846, 66.381, 62.213, 58.321, 54.685,  
01633 51.288, 48.114, 45.145, 42.368, 39.772, 37.341, 35.065, 32.937,  
01634 30.943, 29.077, 27.33, 25.693, 24.158, 22.717, 21.367, 20.099,  
01635 18.909, 17.792, 16.744, 15.761, 14.838, 13.971, 13.157, 12.393,  
01636 11.676, 11.003, 10.369, 9.775, 9.2165, 8.6902, 8.1963, 7.7314,  
01637 7.2923, 6.8794, 6.4898, 6.122, 5.7764, 5.4525, 5.1484, 4.8611,  
01638 4.5918, 4.3379, 4.0982, 3.8716, 3.6567, 3.4545, 3.2634, 3.0828,  
01639 2.9122, 2.7512, 2.5993, 2.4561, 2.3211, 2.1938, 2.0737, 1.9603,  
01640 1.8534, 1.7525, 1.6572, 1.5673, 1.4824, 1.4022, 1.3265, 1.2551,  
01641 1.1876, 1.1239, 1.0637, 1.0069, .9532, .90248, .85454, .80921,  
01642 .76631, .72569, .6872, .65072, .61635, .5836, .55261, .52336,  
01643 .49581, .46998, .44559, .42236, .40036, .37929, .35924, .34043,  
01644 .32238, .30547, .28931, .27405, .25975, .24616, .23341, .22133,  
01645 .20997, .19924, .18917, .17967, .17075, .16211, .15411, .14646,  
01646 .13912, .13201, .12509, .11857, .11261, .10698, .10186, .097039,  
01647 .092236, .087844, .083443, .07938, .075452, .071564, .067931,  
01648 .064389, .061078, .057901, .054921, .052061, .049364, .046789,  
01649 .04435, .042044, .039866, .037808, .035863, .034023, .032282,  
01650 .030634, .029073, .027595, .026194, .024866, .023608, .022415,  
01651 .021283, .02021, .019193, .018228, .017312, .016443, .015619,  
01652 .014837, .014094, .01339, .012721, .012086, .011483, .010911,  
01653 .010368, .009852, .0093623, .0088972, .0084556, .0080362,  
01654 .0076379, .0072596, .0069003, .006559, .0062349, .0059269,  
01655 .0056344, .0053565, .0050925, .0048417, .0046034, .004377,  
01656 .0041618, .0039575, .0037633, .0035788, .0034034, .0032368,

```

01657 .0030785, .002928, .0027851, .0026492, .0025201, .0023975,
01658 .0022809, .0021701, .0020649, .0019649, .0018699, .0017796,
01659 .0016938, .0016122, .0015348, .0014612, .0013913, .001325,
01660 .0012619, .0012021, .0011452, .0010913, .0010401, 9.9149e-4,
01661 9.454e-4, 9.0169e-4, 8.6024e-4, 8.2097e-4, 7.8377e-4, 7.4854e-4,
01662 7.1522e-4, 6.8371e-4, 6.5393e-4, 6.2582e-4, 5.9932e-4, 5.7435e-4,
01663 5.5087e-4, 5.2882e-4, 5.0814e-4, 4.8881e-4, 4.7076e-4, 4.5398e-4,
01664 4.3843e-4, 4.2407e-4, 4.109e-4, 3.9888e-4, 3.88e-4, 3.7826e-4,
01665 3.6963e-4, 3.6213e-4, 3.5575e-4, 3.505e-4, 3.464e-4, 3.4346e-4,
01666 3.4173e-4, 3.4125e-4, 3.4206e-4, 3.4424e-4, 3.4787e-4, 3.5303e-4,
01667 3.5986e-4, 3.6847e-4, 3.7903e-4, 3.9174e-4, 4.0681e-4, 4.2455e-4,
01668 4.4527e-4, 4.6942e-4, 4.9637e-4, 5.2698e-4, 5.5808e-4, 5.9514e-4,
01669 6.2757e-4, 6.689e-4, 7.1298e-4, 7.3955e-4, 7.8403e-4, 8.0449e-4,
01670 8.5131e-4, 9.0256e-4, 9.3692e-4, .0010051, .0010846, .0011678,
01671 .001282, .0014016, .0015355, .0016764, .0018272, .0020055,
01672 .0021455, .0023421, .0024615, .0026786, .0028787, .0031259,
01673 .0034046, .0036985, .0040917, .0043902, .0048349, .0049531,
01674 .0052989, .0056148, .0052452, .0053357, .005333, .0045069,
01675 .0043851, .004253, .003738, .0038084, .0039013, .0041505,
01676 .0045372, .0050569, .0054507, .0061267, .0066122, .0072449,
01677 .0078012, .0082651, .0076538, .0076573, .0076806, .0075227,
01678 .0076269, .0063758, .006254, .0067749, .0067909, .0068231,
01679 .0072143, .0072762, .0072954, .007679, .0075107, .0073658,
01680 .0072441, .0071074, .0070378, .007176, .0072472, .0075844,
01681 .0079291, .008412, .0090165, .010688, .011535, .012375, .013166,
01682 .013895, .015567, .016011, .016392, .016737, .017043, .017731,
01683 .018031, .018419, .018877, .019474, .019868, .020604, .021538,
01684 .022653, .023869, .025288, .026879, .028547, .030524, .03274,
01685 .035132, .03769, .040567, .043793, .047188, .049962, .053542,
01686 .057205, .060776, .061489, .064419, .067124, .065945, .068487,
01687 .071209, .074783, .077039, .082444, .08902, .09692, .10617,
01688 .11687, .12952, .12362, .13498, .14412, .15492, .16519, .1744,
01689 .17096, .17714, .18208, .17363, .17813, .18564, .18295, .19045,
01690 .20252, .20815, .21844, .22929, .24229, .25321, .26588, .2797,
01691 .29465, .31136, .32961, .36529, .38486, .41027, .43694, .4667,
01692 .49943, .54542, .58348, .62303, .67633, .71755, .76054, .81371,
01693 .85934, .90841, .96438, 1.0207, 1.0821, 1.1491, 1.2226, 1.3018,
01694 1.388, 1.4818, 1.5835, 1.6939, 1.8137, 1.9435, 2.0843, 2.237,
01695 2.4026, 2.5818, 2.7767, 2.9885, 3.2182, 3.4679, 3.7391, 4.0349,
01696 4.3554, 4.7053, 5.0849, 5.4986, 5.9436, 6.4294, 6.9598, 7.5203,
01697 8.143, 8.8253, 9.5568, 10.371, 11.267, 12.233, 13.31, 14.357,
01698 15.598, 16.93, 18.358, 19.849, 21.408, 23.04, 24.706, 26.409,
01699 28.153, 28.795, 30.549, 32.43, 34.49, 36.027, 38.955, 42.465,
01700 46.565, 50.875, 55.378, 59.002, 63.882, 67.949, 73.693, 80.095,
01701 86.403, 94.264, 102.65, 112.37, 123.3, 135.54, 149.14, 163.83,
01702 179.17, 196.89, 217.91, 240.94, 264.13, 292.39, 324.83, 358.21,
01703 397.16, 440.5, 488.6, 541.04, 595.3, 650.43, 652.03, 688.74,
01704 719.47, 743.54, 757.68, 762.35, 756.43, 741.42, 595.43, 580.97,
01705 580.83, 605.68, 667.88, 764.49, 759.93, 789.12, 798.17, 645.66,
01706 615.65, 455.05, 421.09, 306.45, 289.14, 235.7, 215.52, 274.57,
01707 316.53, 357.73, 409.89, 465.06, 521.84, 579.02, 630.64, 794.46,
01708 813., 813.56, 796.25, 761.57, 727.97, 812.14, 866.75, 932.5,
01709 1132.8, 1194.8, 1362.2, 1387.2, 1482.3, 1479.7, 1517.9, 1533.1,
01710 1534.2, 1523.3, 1522.5, 1515.5, 1505.2, 1486.5, 1454., 1412.,
01711 1358.8, 1107.8, 1060.9, 1033.5, 1048.2, 1122.4, 1248.9, 1227.1,
01712 1255.4, 1058.9, 1020.7, 970.59, 715.24, 512.56, 468.47, 349.3,
01713 338.26, 299.22, 301.26, 332.38, 382.08, 445.49, 515.87, 590.85,
01714 662.3, 726.05, 955.59, 964.11, 945.17, 891.48, 807.11, 720.9,
01715 803.36, 834.46, 1073.9, 1107.1, 1123.6, 1296., 1393.7, 1303.1,
01716 1284.3, 1161.8, 1078.8, 976.13, 868.72, 767.4, 674.72, 593.73,
01717 523.12, 462.24, 409.75, 364.34, 325., 290.73, 260.76, 234.46,
01718 211.28, 190.78, 172.61, 156.44, 142.01, 129.12, 117.57, 107.2,
01719 97.877, 89.47, 81.882, 75.021, 68.807, 63.171, 58.052, 53.396,
01720 49.155, 45.288, 41.759, 38.531, 35.576, 32.868, 30.384, 28.102,
01721 26.003, 24.071, 22.293, 20.655, 19.147, 17.756, 16.476, 15.292,
01722 14.198, 13.183, 12.241, 11.367, 10.554, 9.7989, 9.0978, 8.4475,
01723 7.845, 7.2868, 6.7704, 6.2927, 5.8508, 5.4421, 5.064, 4.714,
01724 4.3902, 4.0902, 3.8121, 3.5543, 3.315, 3.093, 2.8869, 2.6953,
01725 2.5172, 2.3517, 2.1977, 2.0544, 1.9211, 1.7969, 1.6812, 1.5735,
01726 1.4731, 1.3794, 1.2921, 1.2107, 1.1346, 1.0637, .99744, .93554,
01727 .87771, .82368, .77313, .72587, .6816, .64014, .60134, .565,
01728 .53086, .49883, .46881, .44074, .4144, .38979, .36679, .34513,
01729 .32474, .30552, .28751, .27045, .25458, .23976, .22584, .21278,
01730 .20051, .18899, .17815, .16801, .15846, .14954, .14117, .13328,
01731 .12584
01732 };
01733
01734 /* Get CO2 continuum absorption... */
01735 double xw = nu / 2 + 1;
01736 if (xw >= 1 && xw < 2001) {
01737     int iw = (int) xw;
01738     double dw = xw - iw;
01739     double ew = 1 - dw;
01740     double cw296 = ew * co2296[iw - 1] + dw * co2296[iw];
01741     double cw260 = ew * co2260[iw - 1] + dw * co2260[iw];
01742     double cw230 = ew * co2230[iw - 1] + dw * co2230[iw];
01743     double dt230 = t - 230;

```

```
01744     double dt260 = t - 260;
01745     double dt296 = t - 296;
01746     double ctw = dt260 * 5.050505e-4 * dt296 * cw230 - dt230 * 9.259259e-4
01747         * dt296 * cw260 + dt230 * 4.208754e-4 * dt260 * cw296;
01748     return u / NA / 1000 * p / PO * ctw;
01749 } else
01750     return 0;
01751 }
01752
01753 /*****
01754
01755 double ctmh2o(
01756     double nu,
01757     double p,
01758     double t,
01759     double q,
01760     double u) {
01761
01762     static double h2o296[2001] = { .17, .1695, .172, .168, .1687, .1624, .1606,
01763         .1508, .1447, .1344, .1214, .1133, .1009, .09217, .08297, .06989,
01764         .06513, .05469, .05056, .04417, .03779, .03484, .02994, .0272,
01765         .02325, .02063, .01818, .01592, .01405, .01251, .0108, .009647,
01766         .008424, .007519, .006555, .00588, .005136, .004511, .003989,
01767         .003509, .003114, .00274, .002446, .002144, .001895, .001676,
01768         .001486, .001312, .001164, .001031, 9.129e-4, 8.106e-4, 7.213e-4,
01769         6.4e-4, 5.687e-4, 5.063e-4, 4.511e-4, 4.029e-4, 3.596e-4,
01770         3.22e-4, 2.889e-4, 2.597e-4, 2.337e-4, 2.108e-4, 1.907e-4,
01771         1.728e-4, 1.57e-4, 1.43e-4, 1.305e-4, 1.195e-4, 1.097e-4,
01772         1.009e-4, 9.307e-5, 8.604e-5, 7.971e-5, 7.407e-5, 6.896e-5,
01773         6.433e-5, 6.013e-5, 5.631e-5, 5.283e-5, 4.963e-5, 4.669e-5,
01774         4.398e-5, 4.148e-5, 3.917e-5, 3.702e-5, 3.502e-5, 3.316e-5,
01775         3.142e-5, 2.978e-5, 2.825e-5, 2.681e-5, 2.546e-5, 2.419e-5,
01776         2.299e-5, 2.186e-5, 2.079e-5, 1.979e-5, 1.884e-5, 1.795e-5,
01777         1.711e-5, 1.633e-5, 1.559e-5, 1.49e-5, 1.426e-5, 1.367e-5,
01778         1.312e-5, 1.263e-5, 1.218e-5, 1.178e-5, 1.143e-5, 1.112e-5,
01779         1.088e-5, 1.07e-5, 1.057e-5, 1.05e-5, 1.051e-5, 1.059e-5,
01780         1.076e-5, 1.1e-5, 1.133e-5, 1.18e-5, 1.237e-5, 1.308e-5,
01781         1.393e-5, 1.483e-5, 1.614e-5, 1.758e-5, 1.93e-5, 2.123e-5,
01782         2.346e-5, 2.647e-5, 2.93e-5, 3.279e-5, 3.745e-5, 4.152e-5,
01783         4.813e-5, 5.477e-5, 6.203e-5, 7.331e-5, 8.056e-5, 9.882e-5,
01784         1.05e-4, 1.21e-4, 1.341e-4, 1.572e-4, 1.698e-4, 1.968e-4,
01785         2.175e-4, 2.431e-4, 2.735e-4, 2.867e-4, 3.19e-4, 3.371e-4,
01786         3.554e-4, 3.726e-4, 3.837e-4, 3.878e-4, 3.864e-4, 3.858e-4,
01787         3.841e-4, 3.852e-4, 3.815e-4, 3.762e-4, 3.618e-4, 3.579e-4,
01788         3.45e-4, 3.202e-4, 3.018e-4, 2.785e-4, 2.602e-4, 2.416e-4,
01789         2.097e-4, 1.939e-4, 1.689e-4, 1.498e-4, 1.308e-4, 1.17e-4,
01790         1.011e-4, 9.237e-5, 7.909e-5, 7.006e-5, 6.112e-5, 5.401e-5,
01791         4.914e-5, 4.266e-5, 3.963e-5, 3.316e-5, 3.037e-5, 2.598e-5,
01792         2.294e-5, 2.066e-5, 1.813e-5, 1.583e-5, 1.423e-5, 1.247e-5,
01793         1.116e-5, 9.76e-6, 8.596e-6, 7.72e-6, 6.825e-6, 6.108e-6,
01794         5.366e-6, 4.733e-6, 4.229e-6, 3.731e-6, 3.346e-6, 2.972e-6,
01795         2.628e-6, 2.356e-6, 2.102e-6, 1.878e-6, 1.678e-6, 1.507e-6,
01796         1.348e-6, 1.21e-6, 1.089e-6, 9.806e-7, 8.857e-7, 8.004e-7,
01797         7.261e-7, 6.599e-7, 6.005e-7, 5.479e-7, 5.011e-7, 4.595e-7,
01798         4.219e-7, 3.885e-7, 3.583e-7, 3.314e-7, 3.071e-7, 2.852e-7,
01799         2.654e-7, 2.474e-7, 2.311e-7, 2.162e-7, 2.026e-7, 1.902e-7,
01800         1.788e-7, 1.683e-7, 1.587e-7, 1.497e-7, 1.415e-7, 1.338e-7,
01801         1.266e-7, 1.2e-7, 1.138e-7, 1.08e-7, 1.027e-7, 9.764e-8,
01802         9.296e-8, 8.862e-8, 8.458e-8, 8.087e-8, 7.744e-8, 7.429e-8,
01803         7.145e-8, 6.893e-8, 6.664e-8, 6.468e-8, 6.322e-8, 6.162e-8,
01804         6.07e-8, 5.992e-8, 5.913e-8, 5.841e-8, 5.796e-8, 5.757e-8,
01805         5.746e-8, 5.731e-8, 5.679e-8, 5.577e-8, 5.671e-8, 5.656e-8,
01806         5.594e-8, 5.593e-8, 5.602e-8, 5.62e-8, 5.693e-8, 5.725e-8,
01807         5.858e-8, 6.037e-8, 6.249e-8, 6.535e-8, 6.899e-8, 7.356e-8,
01808         7.918e-8, 8.618e-8, 9.385e-8, 1.039e-7, 1.158e-7, 1.29e-7,
01809         1.437e-7, 1.65e-7, 1.871e-7, 2.121e-7, 2.427e-7, 2.773e-7,
01810         3.247e-7, 3.677e-7, 4.037e-7, 4.776e-7, 5.101e-7, 6.214e-7,
01811         6.936e-7, 7.581e-7, 8.486e-7, 9.355e-7, 9.942e-7, 1.063e-6,
01812         1.123e-6, 1.191e-6, 1.215e-6, 1.247e-6, 1.26e-6, 1.271e-6,
01813         1.284e-6, 1.317e-6, 1.323e-6, 1.349e-6, 1.353e-6, 1.362e-6,
01814         1.344e-6, 1.329e-6, 1.336e-6, 1.327e-6, 1.325e-6, 1.359e-6,
01815         1.374e-6, 1.415e-6, 1.462e-6, 1.526e-6, 1.619e-6, 1.735e-6,
01816         1.863e-6, 2.034e-6, 2.265e-6, 2.482e-6, 2.756e-6, 3.103e-6,
01817         3.466e-6, 3.832e-6, 4.378e-6, 4.913e-6, 5.651e-6, 6.311e-6,
01818         7.169e-6, 8.057e-6, 9.253e-6, 1.047e-5, 1.212e-5, 1.36e-5,
01819         1.569e-5, 1.776e-5, 2.02e-5, 2.281e-5, 2.683e-5, 2.994e-5,
01820         3.488e-5, 3.896e-5, 4.499e-5, 5.175e-5, 6.035e-5, 6.34e-5,
01821         7.281e-5, 7.923e-5, 8.348e-5, 9.631e-5, 1.044e-4, 1.102e-4,
01822         1.176e-4, 1.244e-4, 1.283e-4, 1.326e-4, 1.4e-4, 1.395e-4,
01823         1.387e-4, 1.363e-4, 1.314e-4, 1.241e-4, 1.228e-4, 1.148e-4,
01824         1.086e-4, 1.018e-4, 8.89e-5, 8.316e-5, 7.292e-5, 6.452e-5,
01825         5.625e-5, 5.045e-5, 4.38e-5, 3.762e-5, 3.29e-5, 2.836e-5,
01826         2.485e-5, 2.168e-5, 1.895e-5, 1.659e-5, 1.453e-5, 1.282e-5,
01827         1.132e-5, 1.001e-5, 8.836e-6, 7.804e-6, 6.922e-6, 6.116e-6,
01828         5.429e-6, 4.824e-6, 4.278e-6, 3.788e-6, 3.371e-6, 2.985e-6,
01829         2.649e-6, 2.357e-6, 2.09e-6, 1.858e-6, 1.647e-6, 1.462e-6,
01830         1.299e-6, 1.155e-6, 1.028e-6, 9.142e-7, 8.132e-7, 7.246e-7,
```

```

01831 6.451e-7, 5.764e-7, 5.151e-7, 4.603e-7, 4.121e-7, 3.694e-7,
01832 3.318e-7, 2.985e-7, 2.69e-7, 2.428e-7, 2.197e-7, 1.992e-7,
01833 1.81e-7, 1.649e-7, 1.506e-7, 1.378e-7, 1.265e-7, 1.163e-7,
01834 1.073e-7, 9.918e-8, 9.191e-8, 8.538e-8, 7.949e-8, 7.419e-8,
01835 6.94e-8, 6.508e-8, 6.114e-8, 5.761e-8, 5.437e-8, 5.146e-8,
01836 4.89e-8, 4.636e-8, 4.406e-8, 4.201e-8, 4.015e-8, 3.84e-8,
01837 3.661e-8, 3.51e-8, 3.377e-8, 3.242e-8, 3.13e-8, 3.015e-8,
01838 2.918e-8, 2.83e-8, 2.758e-8, 2.707e-8, 2.656e-8, 2.619e-8,
01839 2.609e-8, 2.615e-8, 2.63e-8, 2.675e-8, 2.745e-8, 2.842e-8,
01840 2.966e-8, 3.125e-8, 3.318e-8, 3.565e-8, 3.85e-8, 4.191e-8,
01841 4.59e-8, 5.059e-8, 5.607e-8, 6.239e-8, 6.958e-8, 7.796e-8,
01842 8.773e-8, 9.88e-8, 1.114e-7, 1.258e-7, 1.422e-7, 1.61e-7,
01843 1.822e-7, 2.06e-7, 2.337e-7, 2.645e-7, 2.996e-7, 3.393e-7,
01844 3.843e-7, 4.363e-7, 4.935e-7, 5.607e-7, 6.363e-7, 7.242e-7,
01845 8.23e-7, 9.411e-7, 1.071e-6, 1.232e-6, 1.402e-6, 1.6e-6, 1.82e-6,
01846 2.128e-6, 2.386e-6, 2.781e-6, 3.242e-6, 3.653e-6, 4.323e-6,
01847 4.747e-6, 5.321e-6, 5.919e-6, 6.681e-6, 7.101e-6, 7.983e-6,
01848 8.342e-6, 8.741e-6, 9.431e-6, 9.952e-6, 1.026e-5, 1.055e-5,
01849 1.095e-5, 1.095e-5, 1.087e-5, 1.056e-5, 1.026e-5, 9.715e-6,
01850 9.252e-6, 8.452e-6, 7.958e-6, 7.268e-6, 6.295e-6, 6.003e-6, 5e-6,
01851 4.591e-6, 3.983e-6, 3.479e-6, 3.058e-6, 2.667e-6, 2.293e-6,
01852 1.995e-6, 1.747e-6, 1.517e-6, 1.335e-6, 1.165e-6, 1.028e-6,
01853 9.007e-7, 7.956e-7, 7.015e-7, 6.192e-7, 5.491e-7, 4.859e-7,
01854 4.297e-7, 3.799e-7, 3.38e-7, 3.002e-7, 2.659e-7, 2.366e-7,
01855 2.103e-7, 1.861e-7, 1.655e-7, 1.469e-7, 1.309e-7, 1.162e-7,
01856 1.032e-7, 9.198e-8, 8.181e-8, 7.294e-8, 6.516e-8, 5.787e-8,
01857 5.163e-8, 4.612e-8, 4.119e-8, 3.695e-8, 3.308e-8, 2.976e-8,
01858 2.67e-8, 2.407e-8, 2.171e-8, 1.965e-8, 1.78e-8, 1.617e-8,
01859 1.47e-8, 1.341e-8, 1.227e-8, 1.125e-8, 1.033e-8, 9.524e-9,
01860 8.797e-9, 8.162e-9, 7.565e-9, 7.04e-9, 6.56e-9, 6.129e-9,
01861 5.733e-9, 5.376e-9, 5.043e-9, 4.75e-9, 4.466e-9, 4.211e-9,
01862 3.977e-9, 3.759e-9, 3.558e-9, 3.373e-9, 3.201e-9, 3.043e-9,
01863 2.895e-9, 2.76e-9, 2.635e-9, 2.518e-9, 2.411e-9, 2.314e-9,
01864 2.23e-9, 2.151e-9, 2.087e-9, 2.035e-9, 1.988e-9, 1.946e-9,
01865 1.927e-9, 1.916e-9, 1.916e-9, 1.933e-9, 1.966e-9, 2.018e-9,
01866 2.09e-9, 2.182e-9, 2.299e-9, 2.442e-9, 2.623e-9, 2.832e-9,
01867 3.079e-9, 3.368e-9, 3.714e-9, 4.104e-9, 4.567e-9, 5.091e-9,
01868 5.701e-9, 6.398e-9, 7.194e-9, 8.127e-9, 9.141e-9, 1.035e-8,
01869 1.177e-8, 1.338e-8, 1.508e-8, 1.711e-8, 1.955e-8, 2.216e-8,
01870 2.534e-8, 2.871e-8, 3.291e-8, 3.711e-8, 4.285e-8, 4.868e-8,
01871 5.509e-8, 6.276e-8, 7.262e-8, 8.252e-8, 9.4e-8, 1.064e-7,
01872 1.247e-7, 1.411e-7, 1.626e-7, 1.827e-7, 2.044e-7, 2.284e-7,
01873 2.452e-7, 2.854e-7, 3.026e-7, 3.278e-7, 3.474e-7, 3.693e-7,
01874 3.93e-7, 4.104e-7, 4.22e-7, 4.439e-7, 4.545e-7, 4.778e-7,
01875 4.812e-7, 5.018e-7, 4.899e-7, 5.075e-7, 5.073e-7, 5.171e-7,
01876 5.131e-7, 5.25e-7, 5.617e-7, 5.846e-7, 6.239e-7, 6.696e-7,
01877 7.398e-7, 8.073e-7, 9.15e-7, 1.009e-6, 1.116e-6, 1.264e-6,
01878 1.439e-6, 1.644e-6, 1.856e-6, 2.147e-6, 2.317e-6, 2.713e-6,
01879 2.882e-6, 2.99e-6, 3.489e-6, 3.581e-6, 4.033e-6, 4.26e-6,
01880 4.543e-6, 4.84e-6, 4.826e-6, 5.013e-6, 5.252e-6, 5.277e-6,
01881 5.306e-6, 5.236e-6, 5.123e-6, 5.171e-6, 4.843e-6, 4.615e-6,
01882 4.385e-6, 3.97e-6, 3.693e-6, 3.231e-6, 2.915e-6, 2.495e-6,
01883 2.144e-6, 1.91e-6, 1.639e-6, 1.417e-6, 1.226e-6, 1.065e-6,
01884 9.29e-7, 8.142e-7, 7.161e-7, 6.318e-7, 5.581e-7, 4.943e-7,
01885 4.376e-7, 3.884e-7, 3.449e-7, 3.06e-7, 2.712e-7, 2.412e-7,
01886 2.139e-7, 1.903e-7, 1.689e-7, 1.499e-7, 1.331e-7, 1.183e-7,
01887 1.05e-7, 9.362e-8, 8.306e-8, 7.403e-8, 6.578e-8, 5.853e-8,
01888 5.216e-8, 4.632e-8, 4.127e-8, 3.678e-8, 3.279e-8, 2.923e-8,
01889 2.612e-8, 2.339e-8, 2.094e-8, 1.877e-8, 1.686e-8, 1.516e-8,
01890 1.366e-8, 1.234e-8, 1.114e-8, 1.012e-8, 9.182e-9, 8.362e-9,
01891 7.634e-9, 6.981e-9, 6.406e-9, 5.888e-9, 5.428e-9, 5.021e-9,
01892 4.65e-9, 4.326e-9, 4.033e-9, 3.77e-9, 3.536e-9, 3.327e-9,
01893 3.141e-9, 2.974e-9, 2.825e-9, 2.697e-9, 2.584e-9, 2.488e-9,
01894 2.406e-9, 2.34e-9, 2.292e-9, 2.259e-9, 2.244e-9, 2.243e-9,
01895 2.272e-9, 2.31e-9, 2.378e-9, 2.454e-9, 2.618e-9, 2.672e-9,
01896 2.831e-9, 3.05e-9, 3.225e-9, 3.425e-9, 3.677e-9, 3.968e-9,
01897 4.221e-9, 4.639e-9, 4.96e-9, 5.359e-9, 5.649e-9, 6.23e-9,
01898 6.716e-9, 7.218e-9, 7.746e-9, 7.988e-9, 8.627e-9, 8.999e-9,
01899 9.442e-9, 9.82e-9, 1.015e-8, 1.06e-8, 1.079e-8, 1.109e-8,
01900 1.137e-8, 1.186e-8, 1.18e-8, 1.187e-8, 1.194e-8, 1.192e-8,
01901 1.224e-8, 1.245e-8, 1.246e-8, 1.318e-8, 1.377e-8, 1.471e-8,
01902 1.582e-8, 1.713e-8, 1.853e-8, 2.063e-8, 2.27e-8, 2.567e-8,
01903 2.891e-8, 3.264e-8, 3.744e-8, 4.286e-8, 4.915e-8, 5.623e-8,
01904 6.336e-8, 7.293e-8, 8.309e-8, 9.319e-8, 1.091e-7, 1.243e-7,
01905 1.348e-7, 1.449e-7, 1.62e-7, 1.846e-7, 1.937e-7, 2.04e-7,
01906 2.179e-7, 2.298e-7, 2.433e-7, 2.439e-7, 2.464e-7, 2.611e-7,
01907 2.617e-7, 2.582e-7, 2.453e-7, 2.401e-7, 2.349e-7, 2.203e-7,
01908 2.066e-7, 1.939e-7, 1.78e-7, 1.558e-7, 1.391e-7, 1.203e-7,
01909 1.048e-7, 9.464e-8, 8.306e-8, 7.239e-8, 6.317e-8, 5.52e-8,
01910 4.847e-8, 4.282e-8, 3.796e-8, 3.377e-8, 2.996e-8, 2.678e-8,
01911 2.4e-8, 2.134e-8, 1.904e-8, 1.705e-8, 1.523e-8, 1.35e-8,
01912 1.204e-8, 1.07e-8, 9.408e-9, 8.476e-9, 7.47e-9, 6.679e-9,
01913 5.929e-9, 5.267e-9, 4.711e-9, 4.172e-9, 3.761e-9, 3.288e-9,
01914 2.929e-9, 2.609e-9, 2.315e-9, 2.042e-9, 1.844e-9, 1.64e-9,
01915 1.47e-9, 1.31e-9, 1.176e-9, 1.049e-9, 9.377e-10, 8.462e-10,
01916 7.616e-10, 6.854e-10, 6.191e-10, 5.596e-10, 5.078e-10, 4.611e-10,
01917 4.197e-10, 3.83e-10, 3.505e-10, 3.215e-10, 2.956e-10, 2.726e-10,

```

01918 2.521e-10, 2.338e-10, 2.173e-10, 2.026e-10, 1.895e-10, 1.777e-10,  
01919 1.672e-10, 1.579e-10, 1.496e-10, 1.423e-10, 1.358e-10, 1.302e-10,  
01920 1.254e-10, 1.216e-10, 1.187e-10, 1.163e-10, 1.147e-10, 1.145e-10,  
01921 1.15e-10, 1.17e-10, 1.192e-10, 1.25e-10, 1.298e-10, 1.345e-10,  
01922 1.405e-10, 1.538e-10, 1.648e-10, 1.721e-10, 1.872e-10, 1.968e-10,  
01923 2.089e-10, 2.172e-10, 2.317e-10, 2.389e-10, 2.503e-10, 2.585e-10,  
01924 2.686e-10, 2.8e-10, 2.895e-10, 3.019e-10, 3.037e-10, 3.076e-10,  
01925 3.146e-10, 3.198e-10, 3.332e-10, 3.397e-10, 3.54e-10, 3.667e-10,  
01926 3.895e-10, 4.071e-10, 4.565e-10, 4.983e-10, 5.439e-10, 5.968e-10,  
01927 6.676e-10, 7.456e-10, 8.405e-10, 9.478e-10, 1.064e-9, 1.218e-9,  
01928 1.386e-9, 1.581e-9, 1.787e-9, 2.032e-9, 2.347e-9, 2.677e-9,  
01929 3.008e-9, 3.544e-9, 4.056e-9, 4.687e-9, 5.331e-9, 6.227e-9,  
01930 6.854e-9, 8.139e-9, 8.945e-9, 9.865e-9, 1.125e-8, 1.178e-8,  
01931 1.364e-8, 1.436e-8, 1.54e-8, 1.672e-8, 1.793e-8, 1.906e-8,  
01932 2.036e-8, 2.144e-8, 2.292e-8, 2.371e-8, 2.493e-8, 2.606e-8,  
01933 2.706e-8, 2.866e-8, 3.036e-8, 3.136e-8, 3.405e-8, 3.665e-8,  
01934 3.837e-8, 4.229e-8, 4.748e-8, 5.32e-8, 5.763e-8, 6.677e-8,  
01935 7.216e-8, 7.716e-8, 8.958e-8, 9.419e-8, 1.036e-7, 1.108e-7,  
01936 1.189e-7, 1.246e-7, 1.348e-7, 1.31e-7, 1.361e-7, 1.364e-7,  
01937 1.363e-7, 1.343e-7, 1.293e-7, 1.254e-7, 1.235e-7, 1.158e-7,  
01938 1.107e-7, 9.961e-8, 9.011e-8, 7.91e-8, 6.916e-8, 6.338e-8,  
01939 5.564e-8, 4.827e-8, 4.198e-8, 3.695e-8, 3.276e-8, 2.929e-8,  
01940 2.633e-8, 2.391e-8, 2.192e-8, 2.021e-8, 1.89e-8, 1.772e-8,  
01941 1.667e-8, 1.603e-8, 1.547e-8, 1.537e-8, 1.492e-8, 1.515e-8,  
01942 1.479e-8, 1.45e-8, 1.513e-8, 1.495e-8, 1.529e-8, 1.565e-8,  
01943 1.564e-8, 1.553e-8, 1.569e-8, 1.584e-8, 1.57e-8, 1.538e-8,  
01944 1.513e-8, 1.472e-8, 1.425e-8, 1.349e-8, 1.328e-8, 1.249e-8,  
01945 1.17e-8, 1.077e-8, 9.514e-9, 8.614e-9, 7.46e-9, 6.621e-9,  
01946 5.775e-9, 5.006e-9, 4.308e-9, 3.747e-9, 3.24e-9, 2.84e-9,  
01947 2.481e-9, 2.184e-9, 1.923e-9, 1.71e-9, 1.504e-9, 1.334e-9,  
01948 1.187e-9, 1.053e-9, 9.367e-10, 8.306e-10, 7.419e-10, 6.63e-10,  
01949 5.918e-10, 5.277e-10, 4.717e-10, 4.222e-10, 3.783e-10, 3.39e-10,  
01950 3.036e-10, 2.729e-10, 2.455e-10, 2.211e-10, 1.995e-10, 1.804e-10,  
01951 1.635e-10, 1.485e-10, 1.355e-10, 1.24e-10, 1.139e-10, 1.051e-10,  
01952 9.757e-11, 9.114e-11, 8.577e-11, 8.139e-11, 7.792e-11, 7.52e-11,  
01953 7.39e-11, 7.311e-11, 7.277e-11, 7.482e-11, 7.698e-11, 8.162e-11,  
01954 8.517e-11, 8.968e-11, 9.905e-11, 1.075e-10, 1.187e-10, 1.291e-10,  
01955 1.426e-10, 1.573e-10, 1.734e-10, 1.905e-10, 2.097e-10, 2.28e-10,  
01956 2.473e-10, 2.718e-10, 2.922e-10, 3.128e-10, 3.361e-10, 3.641e-10,  
01957 3.91e-10, 4.196e-10, 4.501e-10, 4.932e-10, 5.258e-10, 5.755e-10,  
01958 6.253e-10, 6.664e-10, 7.344e-10, 7.985e-10, 8.877e-10, 1.005e-9,  
01959 1.118e-9, 1.251e-9, 1.428e-9, 1.61e-9, 1.888e-9, 2.077e-9,  
01960 2.331e-9, 2.751e-9, 3.061e-9, 3.522e-9, 3.805e-9, 4.181e-9,  
01961 4.575e-9, 5.167e-9, 5.634e-9, 6.007e-9, 6.501e-9, 6.829e-9,  
01962 7.211e-9, 7.262e-9, 7.696e-9, 7.832e-9, 7.799e-9, 7.651e-9,  
01963 7.304e-9, 7.15e-9, 6.977e-9, 6.603e-9, 6.209e-9, 5.69e-9,  
01964 5.432e-9, 4.764e-9, 4.189e-9, 3.64e-9, 3.203e-9, 2.848e-9,  
01965 2.51e-9, 2.194e-9, 1.946e-9, 1.75e-9, 1.567e-9, 1.426e-9,  
01966 1.302e-9, 1.197e-9, 1.109e-9, 1.035e-9, 9.719e-10, 9.207e-10,  
01967 8.957e-10, 8.578e-10, 8.262e-10, 8.117e-10, 7.987e-10, 7.875e-10,  
01968 7.741e-10, 7.762e-10, 7.537e-10, 7.424e-10, 7.474e-10, 7.294e-10,  
01969 7.216e-10, 7.233e-10, 7.075e-10, 6.892e-10, 6.618e-10, 6.314e-10,  
01970 6.208e-10, 5.689e-10, 5.55e-10, 4.984e-10, 4.6e-10, 4.078e-10,  
01971 3.879e-10, 3.459e-10, 2.982e-10, 2.626e-10, 2.329e-10, 1.988e-10,  
01972 1.735e-10, 1.487e-10, 1.297e-10, 1.133e-10, 9.943e-11, 8.736e-11,  
01973 7.726e-11, 6.836e-11, 6.053e-11, 5.384e-11, 4.789e-11, 4.267e-11,  
01974 3.804e-11, 3.398e-11, 3.034e-11, 2.71e-11, 2.425e-11, 2.173e-11,  
01975 1.95e-11, 1.752e-11, 1.574e-11, 1.418e-11, 1.278e-11, 1.154e-11,  
01976 1.044e-11, 9.463e-12, 8.602e-12, 7.841e-12, 7.171e-12, 6.584e-12,  
01977 6.073e-12, 5.631e-12, 5.254e-12, 4.937e-12, 4.679e-12, 4.476e-12,  
01978 4.328e-12, 4.233e-12, 4.194e-12, 4.211e-12, 4.286e-12, 4.424e-12,  
01979 4.628e-12, 4.906e-12, 5.262e-12, 5.708e-12, 6.254e-12, 6.914e-12,  
01980 7.714e-12, 8.677e-12, 9.747e-12, 1.101e-11, 1.256e-11, 1.409e-11,  
01981 1.597e-11, 1.807e-11, 2.034e-11, 2.316e-11, 2.622e-11, 2.962e-11,  
01982 3.369e-11, 3.819e-11, 4.329e-11, 4.932e-11, 5.589e-11, 6.364e-11,  
01983 7.284e-11, 8.236e-11, 9.447e-11, 1.078e-10, 1.229e-10, 1.417e-10,  
01984 1.614e-10, 1.843e-10, 2.107e-10, 2.406e-10, 2.728e-10, 3.195e-10,  
01985 3.595e-10, 4.153e-10, 4.736e-10, 5.41e-10, 6.088e-10, 6.769e-10,  
01986 7.691e-10, 8.545e-10, 9.621e-10, 1.047e-9, 1.161e-9, 1.296e-9,  
01987 1.424e-9, 1.576e-9, 1.739e-9, 1.893e-9, 2.08e-9, 2.336e-9,  
01988 2.604e-9, 2.76e-9, 3.001e-9, 3.365e-9, 3.55e-9, 3.895e-9,  
01989 4.183e-9, 4.614e-9, 4.846e-9, 5.068e-9, 5.427e-9, 5.541e-9,  
01990 5.864e-9, 5.997e-9, 5.997e-9, 6.061e-9, 5.944e-9, 5.855e-9,  
01991 5.661e-9, 5.523e-9, 5.374e-9, 4.94e-9, 4.688e-9, 4.17e-9,  
01992 3.913e-9, 3.423e-9, 2.997e-9, 2.598e-9, 2.253e-9, 1.946e-9,  
01993 1.71e-9, 1.507e-9, 1.336e-9, 1.19e-9, 1.068e-9, 9.623e-10,  
01994 8.772e-10, 8.007e-10, 7.42e-10, 6.884e-10, 6.483e-10, 6.162e-10,  
01995 5.922e-10, 5.688e-10, 5.654e-10, 5.637e-10, 5.701e-10, 5.781e-10,  
01996 5.874e-10, 6.268e-10, 6.357e-10, 6.525e-10, 7.137e-10, 7.441e-10,  
01997 8.024e-10, 8.485e-10, 9.143e-10, 9.536e-10, 9.717e-10, 1.018e-9,  
01998 1.042e-9, 1.054e-9, 1.092e-9, 1.079e-9, 1.064e-9, 1.043e-9,  
01999 1.02e-9, 9.687e-10, 9.273e-10, 9.208e-10, 9.068e-10, 7.687e-10,  
02000 7.385e-10, 6.595e-10, 5.87e-10, 5.144e-10, 4.417e-10, 3.804e-10,  
02001 3.301e-10, 2.866e-10, 2.509e-10, 2.202e-10, 1.947e-10, 1.719e-10,  
02002 1.525e-10, 1.361e-10, 1.21e-10, 1.084e-10, 9.8e-11, 8.801e-11,  
02003 7.954e-11, 7.124e-11, 6.335e-11, 5.76e-11, 5.132e-11, 4.601e-11,  
02004 4.096e-11, 3.657e-11, 3.25e-11, 2.909e-11, 2.587e-11, 2.297e-11,

02005 2.05e-11, 1.828e-11, 1.632e-11, 1.462e-11, 1.314e-11, 1.185e-11,  
02006 1.073e-11, 9.76e-12, 8.922e-12, 8.206e-12, 7.602e-12, 7.1e-12,  
02007 6.694e-12, 6.378e-12, 6.149e-12, 6.004e-12, 5.941e-12, 5.962e-12,  
02008 6.069e-12, 6.265e-12, 6.551e-12, 6.935e-12, 7.457e-12, 8.074e-12,  
02009 8.811e-12, 9.852e-12, 1.086e-11, 1.207e-11, 1.361e-11, 1.553e-11,  
02010 1.737e-11, 1.93e-11, 2.175e-11, 2.41e-11, 2.706e-11, 3.023e-11,  
02011 3.313e-11, 3.657e-11, 4.118e-11, 4.569e-11, 5.025e-11, 5.66e-11,  
02012 6.231e-11, 6.881e-11, 7.996e-11, 8.526e-11, 9.694e-11, 1.106e-10,  
02013 1.222e-10, 1.355e-10, 1.525e-10, 1.775e-10, 1.924e-10, 2.181e-10,  
02014 2.379e-10, 2.662e-10, 2.907e-10, 3.154e-10, 3.366e-10, 3.579e-10,  
02015 3.858e-10, 4.046e-10, 4.196e-10, 4.166e-10, 4.457e-10, 4.466e-10,  
02016 4.404e-10, 4.337e-10, 4.15e-10, 4.083e-10, 3.91e-10, 3.723e-10,  
02017 3.514e-10, 3.303e-10, 2.847e-10, 2.546e-10, 2.23e-10, 1.994e-10,  
02018 1.733e-10, 1.488e-10, 1.297e-10, 1.144e-10, 1.004e-10, 8.741e-11,  
02019 7.928e-11, 7.034e-11, 6.323e-11, 5.754e-11, 5.25e-11, 4.85e-11,  
02020 4.502e-11, 4.286e-11, 4.028e-11, 3.899e-11, 3.824e-11, 3.761e-11,  
02021 3.804e-11, 3.839e-11, 3.845e-11, 4.244e-11, 4.382e-11, 4.582e-11,  
02022 4.847e-11, 5.209e-11, 5.384e-11, 5.887e-11, 6.371e-11, 6.737e-11,  
02023 7.168e-11, 7.415e-11, 7.827e-11, 8.037e-11, 8.12e-11, 8.071e-11,  
02024 8.008e-11, 7.851e-11, 7.544e-11, 7.377e-11, 7.173e-11, 6.801e-11,  
02025 6.267e-11, 5.727e-11, 5.288e-11, 4.853e-11, 4.082e-11, 3.645e-11,  
02026 3.136e-11, 2.672e-11, 2.304e-11, 1.986e-11, 1.725e-11, 1.503e-11,  
02027 1.315e-11, 1.153e-11, 1.014e-11, 8.942e-12, 7.901e-12, 6.993e-12,  
02028 6.199e-12, 5.502e-12, 4.89e-12, 4.351e-12, 3.878e-12, 3.461e-12,  
02029 3.094e-12, 2.771e-12, 2.488e-12, 2.241e-12, 2.025e-12, 1.838e-12,  
02030 1.677e-12, 1.541e-12, 1.427e-12, 1.335e-12, 1.262e-12, 1.209e-12,  
02031 1.176e-12, 1.161e-12, 1.165e-12, 1.189e-12, 1.234e-12, 1.3e-12,  
02032 1.389e-12, 1.503e-12, 1.644e-12, 1.814e-12, 2.017e-12, 2.255e-12,  
02033 2.534e-12, 2.858e-12, 3.231e-12, 3.661e-12, 4.153e-12, 4.717e-12,  
02034 5.36e-12, 6.094e-12, 6.93e-12, 7.882e-12, 8.966e-12, 1.02e-11,  
02035 1.162e-11, 1.324e-11, 1.51e-11, 1.72e-11, 1.965e-11, 2.237e-11,  
02036 2.56e-11, 2.927e-11, 3.371e-11, 3.842e-11, 4.429e-11, 5.139e-11,  
02037 5.798e-11, 6.697e-11, 7.626e-11, 8.647e-11, 1.022e-10, 1.136e-10,  
02038 1.3e-10, 1.481e-10, 1.672e-10, 1.871e-10, 2.126e-10, 2.357e-10,  
02039 2.583e-10, 2.997e-10, 3.289e-10, 3.702e-10, 4.012e-10, 4.319e-10,  
02040 4.527e-10, 5.001e-10, 5.448e-10, 5.611e-10, 5.76e-10, 5.965e-10,  
02041 6.079e-10, 6.207e-10, 6.276e-10, 6.222e-10, 6.137e-10, 6e-10,  
02042 5.814e-10, 5.393e-10, 5.35e-10, 4.947e-10, 4.629e-10, 4.117e-10,  
02043 3.712e-10, 3.372e-10, 2.923e-10, 2.55e-10, 2.232e-10, 1.929e-10,  
02044 1.679e-10, 1.46e-10, 1.289e-10, 1.13e-10, 9.953e-11, 8.763e-11,  
02045 7.76e-11, 6.9e-11, 6.16e-11, 5.525e-11, 4.958e-11, 4.489e-11,  
02046 4.072e-11, 3.728e-11, 3.438e-11, 3.205e-11, 3.006e-11, 2.848e-11,  
02047 2.766e-11, 2.688e-11, 2.664e-11, 2.67e-11, 2.696e-11, 2.786e-11,  
02048 2.861e-11, 3.009e-11, 3.178e-11, 3.389e-11, 3.587e-11, 3.819e-11,  
02049 4.054e-11, 4.417e-11, 4.703e-11, 5.137e-11, 5.46e-11, 6.055e-11,  
02050 6.333e-11, 6.773e-11, 7.219e-11, 7.717e-11, 8.131e-11, 8.491e-11,  
02051 8.574e-11, 9.01e-11, 9.017e-11, 8.999e-11, 8.959e-11, 8.838e-11,  
02052 8.579e-11, 8.162e-11, 8.098e-11, 7.472e-11, 7.108e-11, 6.559e-11,  
02053 5.994e-11, 5.172e-11, 4.424e-11, 3.951e-11, 3.34e-11, 2.902e-11,  
02054 2.541e-11, 2.215e-11, 1.945e-11, 1.716e-11, 1.503e-11, 1.339e-11,  
02055 1.185e-11, 1.05e-11, 9.336e-12, 8.307e-12, 7.312e-12, 6.55e-12,  
02056 5.836e-12, 5.178e-12, 4.6e-12, 4.086e-12, 3.639e-12, 3.247e-12,  
02057 2.904e-12, 2.604e-12, 2.341e-12, 2.112e-12, 1.914e-12, 1.744e-12,  
02058 1.598e-12, 1.476e-12, 1.374e-12, 1.293e-12, 1.23e-12, 1.185e-12,  
02059 1.158e-12, 1.147e-12, 1.154e-12, 1.177e-12, 1.219e-12, 1.28e-12,  
02060 1.36e-12, 1.463e-12, 1.591e-12, 1.75e-12, 1.94e-12, 2.156e-12,  
02061 2.43e-12, 2.748e-12, 3.052e-12, 3.533e-12, 3.967e-12, 4.471e-12,  
02062 5.041e-12, 5.86e-12, 6.664e-12, 7.522e-12, 8.342e-12, 9.412e-12,  
02063 1.072e-11, 1.213e-11, 1.343e-11, 1.496e-11, 1.664e-11, 1.822e-11,  
02064 2.029e-11, 2.233e-11, 2.457e-11, 2.709e-11, 2.928e-11, 3.115e-11,  
02065 3.356e-11, 3.592e-11, 3.818e-11, 3.936e-11, 4.061e-11, 4.149e-11,  
02066 4.299e-11, 4.223e-11, 4.251e-11, 4.287e-11, 4.177e-11, 4.094e-11,  
02067 3.942e-11, 3.772e-11, 3.614e-11, 3.394e-11, 3.222e-11, 2.791e-11,  
02068 2.665e-11, 2.309e-11, 2.032e-11, 1.74e-11, 1.535e-11, 1.323e-11,  
02069 1.151e-11, 9.803e-12, 8.65e-12, 7.54e-12, 6.619e-12, 5.832e-12,  
02070 5.113e-12, 4.503e-12, 3.975e-12, 3.52e-12, 3.112e-12, 2.797e-12,  
02071 2.5e-12, 2.24e-12, 2.013e-12, 1.819e-12, 1.653e-12, 1.513e-12,  
02072 1.395e-12, 1.299e-12, 1.225e-12, 1.168e-12, 1.124e-12, 1.148e-12,  
02073 1.107e-12, 1.128e-12, 1.169e-12, 1.233e-12, 1.307e-12, 1.359e-12,  
02074 1.543e-12, 1.686e-12, 1.794e-12, 2.028e-12, 2.21e-12, 2.441e-12,  
02075 2.653e-12, 2.828e-12, 3.093e-12, 3.28e-12, 3.551e-12, 3.677e-12,  
02076 3.803e-12, 3.844e-12, 4.068e-12, 4.093e-12, 4.002e-12, 3.904e-12,  
02077 3.624e-12, 3.633e-12, 3.622e-12, 3.443e-12, 3.184e-12, 2.934e-12,  
02078 2.476e-12, 2.212e-12, 1.867e-12, 1.594e-12, 1.37e-12, 1.192e-12,  
02079 1.045e-12, 9.211e-13, 8.17e-13, 7.29e-13, 6.55e-13, 5.929e-13,  
02080 5.415e-13, 4.995e-13, 4.661e-13, 4.406e-13, 4.225e-13, 4.116e-13,  
02081 4.075e-13, 4.102e-13, 4.198e-13, 4.365e-13, 4.606e-13, 4.925e-13,  
02082 5.326e-13, 5.818e-13, 6.407e-13, 7.104e-13, 7.92e-13, 8.868e-13,  
02083 9.964e-13, 1.123e-12, 1.268e-12, 1.434e-12, 1.626e-12, 1.848e-12,  
02084 2.107e-12, 2.422e-12, 2.772e-12, 3.145e-12, 3.704e-12, 4.27e-12,  
02085 4.721e-12, 5.361e-12, 6.083e-12, 7.095e-12, 7.968e-12, 9.228e-12,  
02086 1.048e-11, 1.187e-11, 1.336e-11, 1.577e-11, 1.772e-11, 2.017e-11,  
02087 2.25e-11, 2.63e-11, 2.911e-11, 3.356e-11, 3.82e-11, 4.173e-11,  
02088 4.811e-11, 5.254e-11, 5.839e-11, 6.187e-11, 6.805e-11, 7.118e-11,  
02089 7.369e-11, 7.664e-11, 7.794e-11, 7.947e-11, 8.036e-11, 7.954e-11,  
02090 7.849e-11, 7.518e-11, 7.462e-11, 6.926e-11, 6.531e-11, 6.197e-11,  
02091 5.421e-11, 4.777e-11, 4.111e-11, 3.679e-11, 3.166e-11, 2.786e-11,



```
02092      2.436e-11, 2.144e-11, 1.859e-11, 1.628e-11, 1.414e-11, 1.237e-11,
02093      1.093e-11, 9.558e-12
02094  };
02095
02096  static double h2o260[2001] = { .2752, .2732, .2749, .2676, .2667, .2545,
02097      .2497, .2327, .2218, .2036, .1825, .1694, .1497, .1353, .121,
02098      .1014, .09405, .07848, .07195, .06246, .05306, .04853, .04138,
02099      .03735, .03171, .02785, .02431, .02111, .01845, .0164, .01405,
02100      .01255, .01098, .009797, .008646, .007779, .006898, .006099,
02101      .005453, .004909, .004413, .003959, .003581, .003199, .002871,
02102      .002583, .00233, .002086, .001874, .001684, .001512, .001361,
02103      .001225, .0011, 9.89e-4, 8.916e-4, 8.039e-4, 7.256e-4, 6.545e-4,
02104      5.918e-4, 5.359e-4, 4.867e-4, 4.426e-4, 4.033e-4, 3.682e-4,
02105      3.366e-4, 3.085e-4, 2.833e-4, 2.605e-4, 2.403e-4, 2.221e-4,
02106      2.055e-4, 1.908e-4, 1.774e-4, 1.653e-4, 1.544e-4, 1.443e-4,
02107      1.351e-4, 1.267e-4, 1.19e-4, 1.119e-4, 1.053e-4, 9.922e-5,
02108      9.355e-5, 8.831e-5, 8.339e-5, 7.878e-5, 7.449e-5, 7.043e-5,
02109      6.664e-5, 6.307e-5, 5.969e-5, 5.654e-5, 5.357e-5, 5.075e-5,
02110      4.81e-5, 4.56e-5, 4.322e-5, 4.102e-5, 3.892e-5, 3.696e-5,
02111      3.511e-5, 3.339e-5, 3.177e-5, 3.026e-5, 2.886e-5, 2.756e-5,
02112      2.636e-5, 2.527e-5, 2.427e-5, 2.337e-5, 2.257e-5, 2.185e-5,
02113      2.127e-5, 2.08e-5, 2.041e-5, 2.013e-5, 2e-5, 1.997e-5, 2.009e-5,
02114      2.031e-5, 2.068e-5, 2.124e-5, 2.189e-5, 2.267e-5, 2.364e-5,
02115      2.463e-5, 2.618e-5, 2.774e-5, 2.937e-5, 3.144e-5, 3.359e-5,
02116      3.695e-5, 4.002e-5, 4.374e-5, 4.947e-5, 5.431e-5, 6.281e-5,
02117      7.169e-5, 8.157e-5, 9.728e-5, 1.079e-4, 1.337e-4, 1.442e-4,
02118      1.683e-4, 1.879e-4, 2.223e-4, 2.425e-4, 2.838e-4, 3.143e-4,
02119      3.527e-4, 4.012e-4, 4.237e-4, 4.747e-4, 5.057e-4, 5.409e-4,
02120      5.734e-4, 5.944e-4, 6.077e-4, 6.175e-4, 6.238e-4, 6.226e-4,
02121      6.248e-4, 6.192e-4, 6.098e-4, 5.818e-4, 5.709e-4, 5.465e-4,
02122      5.043e-4, 4.699e-4, 4.294e-4, 3.984e-4, 3.672e-4, 3.152e-4,
02123      2.883e-4, 2.503e-4, 2.211e-4, 1.92e-4, 1.714e-4, 1.485e-4,
02124      1.358e-4, 1.156e-4, 1.021e-4, 8.887e-5, 7.842e-5, 7.12e-5,
02125      6.186e-5, 5.73e-5, 4.792e-5, 4.364e-5, 3.72e-5, 3.28e-5,
02126      2.946e-5, 2.591e-5, 2.261e-5, 2.048e-5, 1.813e-5, 1.63e-5,
02127      1.447e-5, 1.282e-5, 1.167e-5, 1.041e-5, 9.449e-6, 8.51e-6,
02128      7.596e-6, 6.961e-6, 6.272e-6, 5.728e-6, 5.198e-6, 4.667e-6,
02129      4.288e-6, 3.897e-6, 3.551e-6, 3.235e-6, 2.952e-6, 2.688e-6,
02130      2.449e-6, 2.241e-6, 2.05e-6, 1.879e-6, 1.722e-6, 1.582e-6,
02131      1.456e-6, 1.339e-6, 1.236e-6, 1.144e-6, 1.06e-6, 9.83e-7,
02132      9.149e-7, 8.535e-7, 7.973e-7, 7.466e-7, 6.999e-7, 6.574e-7,
02133      6.18e-7, 5.821e-7, 5.487e-7, 5.18e-7, 4.896e-7, 4.631e-7,
02134      4.386e-7, 4.16e-7, 3.945e-7, 3.748e-7, 3.562e-7, 3.385e-7,
02135      3.222e-7, 3.068e-7, 2.922e-7, 2.788e-7, 2.659e-7, 2.539e-7,
02136      2.425e-7, 2.318e-7, 2.219e-7, 2.127e-7, 2.039e-7, 1.958e-7,
02137      1.885e-7, 1.818e-7, 1.758e-7, 1.711e-7, 1.662e-7, 1.63e-7,
02138      1.605e-7, 1.58e-7, 1.559e-7, 1.545e-7, 1.532e-7, 1.522e-7,
02139      1.51e-7, 1.495e-7, 1.465e-7, 1.483e-7, 1.469e-7, 1.448e-7,
02140      1.444e-7, 1.436e-7, 1.426e-7, 1.431e-7, 1.425e-7, 1.445e-7,
02141      1.477e-7, 1.515e-7, 1.567e-7, 1.634e-7, 1.712e-7, 1.802e-7,
02142      1.914e-7, 2.024e-7, 2.159e-7, 2.295e-7, 2.461e-7, 2.621e-7,
02143      2.868e-7, 3.102e-7, 3.394e-7, 3.784e-7, 4.223e-7, 4.864e-7,
02144      5.501e-7, 6.039e-7, 7.193e-7, 7.728e-7, 9.514e-7, 1.073e-6,
02145      1.18e-6, 1.333e-6, 1.472e-6, 1.566e-6, 1.677e-6, 1.784e-6,
02146      1.904e-6, 1.953e-6, 2.02e-6, 2.074e-6, 2.128e-6, 2.162e-6,
02147      2.219e-6, 2.221e-6, 2.249e-6, 2.239e-6, 2.235e-6, 2.185e-6,
02148      2.141e-6, 2.124e-6, 2.09e-6, 2.068e-6, 2.1e-6, 2.104e-6,
02149      2.142e-6, 2.181e-6, 2.257e-6, 2.362e-6, 2.5e-6, 2.664e-6,
02150      2.884e-6, 3.189e-6, 3.48e-6, 3.847e-6, 4.313e-6, 4.79e-6,
02151      5.25e-6, 5.989e-6, 6.692e-6, 7.668e-6, 8.52e-6, 9.606e-6,
02152      1.073e-5, 1.225e-5, 1.377e-5, 1.582e-5, 1.761e-5, 2.029e-5,
02153      2.284e-5, 2.602e-5, 2.94e-5, 3.483e-5, 3.928e-5, 4.618e-5,
02154      5.24e-5, 6.132e-5, 7.183e-5, 8.521e-5, 9.111e-5, 1.07e-4,
02155      1.184e-4, 1.264e-4, 1.475e-4, 1.612e-4, 1.704e-4, 1.818e-4,
02156      1.924e-4, 1.994e-4, 2.061e-4, 2.18e-4, 2.187e-4, 2.2e-4,
02157      2.196e-4, 2.131e-4, 2.015e-4, 1.988e-4, 1.847e-4, 1.729e-4,
02158      1.597e-4, 1.373e-4, 1.262e-4, 1.087e-4, 9.439e-5, 8.061e-5,
02159      7.093e-5, 6.049e-5, 5.12e-5, 4.435e-5, 3.817e-5, 3.34e-5,
02160      2.927e-5, 2.573e-5, 2.291e-5, 2.04e-5, 1.827e-5, 1.636e-5,
02161      1.463e-5, 1.309e-5, 1.17e-5, 1.047e-5, 9.315e-6, 8.328e-6,
02162      7.458e-6, 6.665e-6, 5.94e-6, 5.316e-6, 4.752e-6, 4.252e-6,
02163      3.825e-6, 3.421e-6, 3.064e-6, 2.746e-6, 2.465e-6, 2.216e-6,
02164      1.99e-6, 1.79e-6, 1.609e-6, 1.449e-6, 1.306e-6, 1.177e-6,
02165      1.063e-6, 9.607e-7, 8.672e-7, 7.855e-7, 7.118e-7, 6.46e-7,
02166      5.871e-7, 5.34e-7, 4.868e-7, 4.447e-7, 4.068e-7, 3.729e-7,
02167      3.423e-7, 3.151e-7, 2.905e-7, 2.686e-7, 2.484e-7, 2.306e-7,
02168      2.142e-7, 1.995e-7, 1.86e-7, 1.738e-7, 1.626e-7, 1.522e-7,
02169      1.427e-7, 1.338e-7, 1.258e-7, 1.183e-7, 1.116e-7, 1.056e-7,
02170      9.972e-8, 9.46e-8, 9.007e-8, 8.592e-8, 8.195e-8, 7.816e-8,
02171      7.483e-8, 7.193e-8, 6.892e-8, 6.642e-8, 6.386e-8, 6.154e-8,
02172      5.949e-8, 5.764e-8, 5.622e-8, 5.479e-8, 5.364e-8, 5.301e-8,
02173      5.267e-8, 5.263e-8, 5.313e-8, 5.41e-8, 5.55e-8, 5.745e-8,
02174      6.003e-8, 6.311e-8, 6.713e-8, 7.173e-8, 7.724e-8, 8.368e-8,
02175      9.121e-8, 9.986e-8, 1.097e-7, 1.209e-7, 1.338e-7, 1.486e-7,
02176      1.651e-7, 1.837e-7, 2.048e-7, 2.289e-7, 2.557e-7, 2.857e-7,
02177      3.195e-7, 3.587e-7, 4.015e-7, 4.497e-7, 5.049e-7, 5.665e-7,
02178      6.366e-7, 7.121e-7, 7.996e-7, 8.946e-7, 1.002e-6, 1.117e-6,
```

02179 1.262e-6, 1.416e-6, 1.611e-6, 1.807e-6, 2.056e-6, 2.351e-6,  
02180 2.769e-6, 3.138e-6, 3.699e-6, 4.386e-6, 5.041e-6, 6.074e-6,  
02181 6.812e-6, 7.79e-6, 8.855e-6, 1.014e-5, 1.095e-5, 1.245e-5,  
02182 1.316e-5, 1.39e-5, 1.504e-5, 1.583e-5, 1.617e-5, 1.652e-5,  
02183 1.713e-5, 1.724e-5, 1.715e-5, 1.668e-5, 1.629e-5, 1.552e-5,  
02184 1.478e-5, 1.34e-5, 1.245e-5, 1.121e-5, 9.575e-6, 8.956e-6,  
02185 7.345e-6, 6.597e-6, 5.612e-6, 4.818e-6, 4.165e-6, 3.579e-6,  
02186 3.041e-6, 2.623e-6, 2.29e-6, 1.984e-6, 1.748e-6, 1.534e-6,  
02187 1.369e-6, 1.219e-6, 1.092e-6, 9.8e-7, 8.762e-7, 7.896e-7,  
02188 7.104e-7, 6.364e-7, 5.691e-7, 5.107e-7, 4.575e-7, 4.09e-7,  
02189 3.667e-7, 3.287e-7, 2.931e-7, 2.633e-7, 2.356e-7, 2.111e-7,  
02190 1.895e-7, 1.697e-7, 1.525e-7, 1.369e-7, 1.233e-7, 1.114e-7,  
02191 9.988e-8, 9.004e-8, 8.149e-8, 7.352e-8, 6.662e-8, 6.03e-8,  
02192 5.479e-8, 4.974e-8, 4.532e-8, 4.129e-8, 3.781e-8, 3.462e-8,  
02193 3.176e-8, 2.919e-8, 2.687e-8, 2.481e-8, 2.292e-8, 2.119e-8,  
02194 1.967e-8, 1.828e-8, 1.706e-8, 1.589e-8, 1.487e-8, 1.393e-8,  
02195 1.307e-8, 1.228e-8, 1.156e-8, 1.089e-8, 1.028e-8, 9.696e-9,  
02196 9.159e-9, 8.658e-9, 8.187e-9, 7.746e-9, 7.34e-9, 6.953e-9,  
02197 6.594e-9, 6.259e-9, 5.948e-9, 5.66e-9, 5.386e-9, 5.135e-9,  
02198 4.903e-9, 4.703e-9, 4.515e-9, 4.362e-9, 4.233e-9, 4.117e-9,  
02199 4.017e-9, 3.962e-9, 3.924e-9, 3.905e-9, 3.922e-9, 3.967e-9,  
02200 4.046e-9, 4.165e-9, 4.32e-9, 4.522e-9, 4.769e-9, 5.083e-9,  
02201 5.443e-9, 5.872e-9, 6.366e-9, 6.949e-9, 7.601e-9, 8.371e-9,  
02202 9.22e-9, 1.02e-8, 1.129e-8, 1.251e-8, 1.393e-8, 1.542e-8,  
02203 1.72e-8, 1.926e-8, 2.152e-8, 2.392e-8, 2.678e-8, 3.028e-8,  
02204 3.39e-8, 3.836e-8, 4.309e-8, 4.9e-8, 5.481e-8, 6.252e-8,  
02205 7.039e-8, 7.883e-8, 8.849e-8, 1.012e-7, 1.142e-7, 1.3e-7,  
02206 1.475e-7, 1.732e-7, 1.978e-7, 2.304e-7, 2.631e-7, 2.988e-7,  
02207 3.392e-7, 3.69e-7, 4.355e-7, 4.672e-7, 5.11e-7, 5.461e-7,  
02208 5.828e-7, 6.233e-7, 6.509e-7, 6.672e-7, 6.969e-7, 7.104e-7,  
02209 7.439e-7, 7.463e-7, 7.708e-7, 7.466e-7, 7.668e-7, 7.549e-7,  
02210 7.586e-7, 7.384e-7, 7.439e-7, 7.785e-7, 7.915e-7, 8.31e-7,  
02211 8.745e-7, 9.558e-7, 1.038e-6, 1.173e-6, 1.304e-6, 1.452e-6,  
02212 1.671e-6, 1.931e-6, 2.239e-6, 2.578e-6, 3.032e-6, 3.334e-6,  
02213 3.98e-6, 4.3e-6, 4.518e-6, 5.321e-6, 5.508e-6, 6.211e-6, 6.59e-6,  
02214 7.046e-6, 7.555e-6, 7.558e-6, 7.875e-6, 8.319e-6, 8.433e-6,  
02215 8.59e-6, 8.503e-6, 8.304e-6, 8.336e-6, 7.739e-6, 7.301e-6,  
02216 6.827e-6, 6.078e-6, 5.551e-6, 4.762e-6, 4.224e-6, 3.538e-6,  
02217 2.984e-6, 2.619e-6, 2.227e-6, 1.923e-6, 1.669e-6, 1.462e-6,  
02218 1.294e-6, 1.155e-6, 1.033e-6, 9.231e-7, 8.238e-7, 7.36e-7,  
02219 6.564e-7, 5.869e-7, 5.236e-7, 4.673e-7, 4.174e-7, 3.736e-7,  
02220 3.33e-7, 2.976e-7, 2.657e-7, 2.367e-7, 2.106e-7, 1.877e-7,  
02221 1.671e-7, 1.494e-7, 1.332e-7, 1.192e-7, 1.065e-7, 9.558e-8,  
02222 8.586e-8, 7.717e-8, 6.958e-8, 6.278e-8, 5.666e-8, 5.121e-8,  
02223 4.647e-8, 4.213e-8, 3.815e-8, 3.459e-8, 3.146e-8, 2.862e-8,  
02224 2.604e-8, 2.375e-8, 2.162e-8, 1.981e-8, 1.817e-8, 1.67e-8,  
02225 1.537e-8, 1.417e-8, 1.31e-8, 1.215e-8, 1.128e-8, 1.05e-8,  
02226 9.793e-9, 9.158e-9, 8.586e-9, 8.068e-9, 7.595e-9, 7.166e-9,  
02227 6.778e-9, 6.427e-9, 6.108e-9, 5.826e-9, 5.571e-9, 5.347e-9,  
02228 5.144e-9, 4.968e-9, 4.822e-9, 4.692e-9, 4.589e-9, 4.506e-9,  
02229 4.467e-9, 4.44e-9, 4.466e-9, 4.515e-9, 4.718e-9, 4.729e-9,  
02230 4.937e-9, 5.249e-9, 5.466e-9, 5.713e-9, 6.03e-9, 6.436e-9,  
02231 6.741e-9, 7.33e-9, 7.787e-9, 8.414e-9, 8.908e-9, 9.868e-9,  
02232 1.069e-8, 1.158e-8, 1.253e-8, 1.3e-8, 1.409e-8, 1.47e-8,  
02233 1.548e-8, 1.612e-8, 1.666e-8, 1.736e-8, 1.763e-8, 1.812e-8,  
02234 1.852e-8, 1.923e-8, 1.897e-8, 1.893e-8, 1.888e-8, 1.868e-8,  
02235 1.895e-8, 1.899e-8, 1.876e-8, 1.96e-8, 2.02e-8, 2.121e-8,  
02236 2.239e-8, 2.379e-8, 2.526e-8, 2.766e-8, 2.994e-8, 3.332e-8,  
02237 3.703e-8, 4.158e-8, 4.774e-8, 5.499e-8, 6.355e-8, 7.349e-8,  
02238 8.414e-8, 9.846e-8, 1.143e-7, 1.307e-7, 1.562e-7, 1.817e-7,  
02239 2.011e-7, 2.192e-7, 2.485e-7, 2.867e-7, 3.035e-7, 3.223e-7,  
02240 3.443e-7, 3.617e-7, 3.793e-7, 3.793e-7, 3.839e-7, 4.081e-7,  
02241 4.117e-7, 4.085e-7, 3.92e-7, 3.851e-7, 3.754e-7, 3.49e-7,  
02242 3.229e-7, 2.978e-7, 2.691e-7, 2.312e-7, 2.029e-7, 1.721e-7,  
02243 1.472e-7, 1.308e-7, 1.132e-7, 9.736e-8, 8.458e-8, 7.402e-8,  
02244 6.534e-8, 5.811e-8, 5.235e-8, 4.762e-8, 4.293e-8, 3.896e-8,  
02245 3.526e-8, 3.165e-8, 2.833e-8, 2.551e-8, 2.288e-8, 2.036e-8,  
02246 1.82e-8, 1.626e-8, 1.438e-8, 1.299e-8, 1.149e-8, 1.03e-8,  
02247 9.148e-9, 8.122e-9, 7.264e-9, 6.425e-9, 5.777e-9, 5.06e-9,  
02248 4.502e-9, 4.013e-9, 3.567e-9, 3.145e-9, 2.864e-9, 2.553e-9,  
02249 2.311e-9, 2.087e-9, 1.886e-9, 1.716e-9, 1.556e-9, 1.432e-9,  
02250 1.311e-9, 1.202e-9, 1.104e-9, 1.013e-9, 9.293e-10, 8.493e-10,  
02251 7.79e-10, 7.185e-10, 6.642e-10, 6.141e-10, 5.684e-10, 5.346e-10,  
02252 5.032e-10, 4.725e-10, 4.439e-10, 4.176e-10, 3.93e-10, 3.714e-10,  
02253 3.515e-10, 3.332e-10, 3.167e-10, 3.02e-10, 2.887e-10, 2.769e-10,  
02254 2.665e-10, 2.578e-10, 2.503e-10, 2.436e-10, 2.377e-10, 2.342e-10,  
02255 2.305e-10, 2.296e-10, 2.278e-10, 2.321e-10, 2.355e-10, 2.402e-10,  
02256 2.478e-10, 2.67e-10, 2.848e-10, 2.982e-10, 3.263e-10, 3.438e-10,  
02257 3.649e-10, 3.829e-10, 4.115e-10, 4.264e-10, 4.473e-10, 4.63e-10,  
02258 4.808e-10, 4.995e-10, 5.142e-10, 5.313e-10, 5.318e-10, 5.358e-10,  
02259 5.452e-10, 5.507e-10, 5.698e-10, 5.782e-10, 5.983e-10, 6.164e-10,  
02260 6.532e-10, 6.811e-10, 7.624e-10, 8.302e-10, 9.067e-10, 9.937e-10,  
02261 1.104e-9, 1.221e-9, 1.361e-9, 1.516e-9, 1.675e-9, 1.883e-9,  
02262 2.101e-9, 2.349e-9, 2.614e-9, 2.92e-9, 3.305e-9, 3.724e-9,  
02263 4.142e-9, 4.887e-9, 5.614e-9, 6.506e-9, 7.463e-9, 8.817e-9,  
02264 9.849e-9, 1.187e-8, 1.321e-8, 1.474e-8, 1.698e-8, 1.794e-8,  
02265 2.09e-8, 2.211e-8, 2.362e-8, 2.556e-8, 2.729e-8, 2.88e-8,

02266 3.046e-8, 3.167e-8, 3.367e-8, 3.457e-8, 3.59e-8, 3.711e-8,  
02267 3.826e-8, 4.001e-8, 4.211e-8, 4.315e-8, 4.661e-8, 5.01e-8,  
02268 5.249e-8, 5.84e-8, 6.628e-8, 7.512e-8, 8.253e-8, 9.722e-8,  
02269 1.067e-7, 1.153e-7, 1.347e-7, 1.428e-7, 1.577e-7, 1.694e-7,  
02270 1.833e-7, 1.938e-7, 2.108e-7, 2.059e-7, 2.157e-7, 2.185e-7,  
02271 2.208e-7, 2.182e-7, 2.093e-7, 2.014e-7, 1.962e-7, 1.819e-7,  
02272 1.713e-7, 1.51e-7, 1.34e-7, 1.154e-7, 9.89e-8, 8.88e-8, 7.673e-8,  
02273 6.599e-8, 5.73e-8, 5.081e-8, 4.567e-8, 4.147e-8, 3.773e-8,  
02274 3.46e-8, 3.194e-8, 2.953e-8, 2.759e-8, 2.594e-8, 2.442e-8,  
02275 2.355e-8, 2.283e-8, 2.279e-8, 2.231e-8, 2.279e-8, 2.239e-8,  
02276 2.21e-8, 2.309e-8, 2.293e-8, 2.352e-8, 2.415e-8, 2.43e-8,  
02277 2.426e-8, 2.465e-8, 2.5e-8, 2.496e-8, 2.465e-8, 2.445e-8,  
02278 2.383e-8, 2.299e-8, 2.165e-8, 2.113e-8, 1.968e-8, 1.819e-8,  
02279 1.644e-8, 1.427e-8, 1.27e-8, 1.082e-8, 9.428e-9, 8.091e-9,  
02280 6.958e-9, 5.988e-9, 5.246e-9, 4.601e-9, 4.098e-9, 3.664e-9,  
02281 3.287e-9, 2.942e-9, 2.656e-9, 2.364e-9, 2.118e-9, 1.903e-9,  
02282 1.703e-9, 1.525e-9, 1.365e-9, 1.229e-9, 1.107e-9, 9.96e-10,  
02283 8.945e-10, 8.08e-10, 7.308e-10, 6.616e-10, 5.994e-10, 5.422e-10,  
02284 4.929e-10, 4.478e-10, 4.07e-10, 3.707e-10, 3.379e-10, 3.087e-10,  
02285 2.823e-10, 2.592e-10, 2.385e-10, 2.201e-10, 2.038e-10, 1.897e-10,  
02286 1.774e-10, 1.667e-10, 1.577e-10, 1.502e-10, 1.437e-10, 1.394e-10,  
02287 1.358e-10, 1.324e-10, 1.329e-10, 1.324e-10, 1.36e-10, 1.39e-10,  
02288 1.424e-10, 1.544e-10, 1.651e-10, 1.817e-10, 1.984e-10, 2.195e-10,  
02289 2.438e-10, 2.7e-10, 2.991e-10, 3.322e-10, 3.632e-10, 3.957e-10,  
02290 4.36e-10, 4.701e-10, 5.03e-10, 5.381e-10, 5.793e-10, 6.19e-10,  
02291 6.596e-10, 7.004e-10, 7.561e-10, 7.934e-10, 8.552e-10, 9.142e-10,  
02292 9.57e-10, 1.027e-9, 1.097e-9, 1.193e-9, 1.334e-9, 1.47e-9,  
02293 1.636e-9, 1.871e-9, 2.122e-9, 2.519e-9, 2.806e-9, 3.203e-9,  
02294 3.846e-9, 4.362e-9, 5.114e-9, 5.643e-9, 6.305e-9, 6.981e-9,  
02295 7.983e-9, 8.783e-9, 9.419e-9, 1.017e-8, 1.063e-8, 1.121e-8,  
02296 1.13e-8, 1.201e-8, 1.225e-8, 1.232e-8, 1.223e-8, 1.177e-8,  
02297 1.151e-8, 1.116e-8, 1.047e-8, 9.698e-9, 8.734e-9, 8.202e-9,  
02298 7.041e-9, 6.074e-9, 5.172e-9, 4.468e-9, 3.913e-9, 3.414e-9,  
02299 2.975e-9, 2.65e-9, 2.406e-9, 2.173e-9, 2.009e-9, 1.861e-9,  
02300 1.727e-9, 1.612e-9, 1.514e-9, 1.43e-9, 1.362e-9, 1.333e-9,  
02301 1.288e-9, 1.249e-9, 1.238e-9, 1.228e-9, 1.217e-9, 1.202e-9,  
02302 1.209e-9, 1.177e-9, 1.157e-9, 1.165e-9, 1.142e-9, 1.131e-9,  
02303 1.138e-9, 1.117e-9, 1.1e-9, 1.069e-9, 1.023e-9, 1.005e-9,  
02304 9.159e-10, 8.863e-10, 7.865e-10, 7.153e-10, 6.247e-10, 5.846e-10,  
02305 5.133e-10, 4.36e-10, 3.789e-10, 3.335e-10, 2.833e-10, 2.483e-10,  
02306 2.155e-10, 1.918e-10, 1.709e-10, 1.529e-10, 1.374e-10, 1.235e-10,  
02307 1.108e-10, 9.933e-11, 8.932e-11, 8.022e-11, 7.224e-11, 6.52e-11,  
02308 5.896e-11, 5.328e-11, 4.813e-11, 4.365e-11, 3.961e-11, 3.594e-11,  
02309 3.266e-11, 2.967e-11, 2.701e-11, 2.464e-11, 2.248e-11, 2.054e-11,  
02310 1.878e-11, 1.721e-11, 1.579e-11, 1.453e-11, 1.341e-11, 1.241e-11,  
02311 1.154e-11, 1.078e-11, 1.014e-11, 9.601e-12, 9.167e-12, 8.838e-12,  
02312 8.614e-12, 8.493e-12, 8.481e-12, 8.581e-12, 8.795e-12, 9.131e-12,  
02313 9.601e-12, 1.021e-11, 1.097e-11, 1.191e-11, 1.303e-11, 1.439e-11,  
02314 1.601e-11, 1.778e-11, 1.984e-11, 2.234e-11, 2.474e-11, 2.766e-11,  
02315 3.085e-11, 3.415e-11, 3.821e-11, 4.261e-11, 4.748e-11, 5.323e-11,  
02316 5.935e-11, 6.619e-11, 7.418e-11, 8.294e-11, 9.26e-11, 1.039e-10,  
02317 1.156e-10, 1.297e-10, 1.46e-10, 1.641e-10, 1.858e-10, 2.1e-10,  
02318 2.383e-10, 2.724e-10, 3.116e-10, 3.538e-10, 4.173e-10, 4.727e-10,  
02319 5.503e-10, 6.337e-10, 7.32e-10, 8.298e-10, 9.328e-10, 1.059e-9,  
02320 1.176e-9, 1.328e-9, 1.445e-9, 1.593e-9, 1.77e-9, 1.954e-9,  
02321 2.175e-9, 2.405e-9, 2.622e-9, 2.906e-9, 3.294e-9, 3.713e-9,  
02322 3.98e-9, 4.384e-9, 4.987e-9, 5.311e-9, 5.874e-9, 6.337e-9,  
02323 7.027e-9, 7.39e-9, 7.769e-9, 8.374e-9, 8.605e-9, 9.165e-9,  
02324 9.415e-9, 9.511e-9, 9.704e-9, 9.588e-9, 9.45e-9, 9.086e-9,  
02325 8.798e-9, 8.469e-9, 7.697e-9, 7.168e-9, 6.255e-9, 5.772e-9,  
02326 4.97e-9, 4.271e-9, 3.653e-9, 3.154e-9, 2.742e-9, 2.435e-9,  
02327 2.166e-9, 1.936e-9, 1.731e-9, 1.556e-9, 1.399e-9, 1.272e-9,  
02328 1.157e-9, 1.066e-9, 9.844e-10, 9.258e-10, 8.787e-10, 8.421e-10,  
02329 8.083e-10, 8.046e-10, 8.067e-10, 8.181e-10, 8.325e-10, 8.517e-10,  
02330 9.151e-10, 9.351e-10, 9.677e-10, 1.071e-9, 1.126e-9, 1.219e-9,  
02331 1.297e-9, 1.408e-9, 1.476e-9, 1.517e-9, 1.6e-9, 1.649e-9,  
02332 1.678e-9, 1.746e-9, 1.742e-9, 1.728e-9, 1.699e-9, 1.655e-9,  
02333 1.561e-9, 1.48e-9, 1.451e-9, 1.411e-9, 1.171e-9, 1.106e-9,  
02334 9.714e-10, 8.523e-10, 7.346e-10, 6.241e-10, 5.371e-10, 4.704e-10,  
02335 4.144e-10, 3.683e-10, 3.292e-10, 2.942e-10, 2.62e-10, 2.341e-10,  
02336 2.104e-10, 1.884e-10, 1.7e-10, 1.546e-10, 1.394e-10, 1.265e-10,  
02337 1.14e-10, 1.019e-10, 9.279e-11, 8.283e-11, 7.458e-11, 6.668e-11,  
02338 5.976e-11, 5.33e-11, 4.794e-11, 4.289e-11, 3.841e-11, 3.467e-11,  
02339 3.13e-11, 2.832e-11, 2.582e-11, 2.356e-11, 2.152e-11, 1.97e-11,  
02340 1.808e-11, 1.664e-11, 1.539e-11, 1.434e-11, 1.344e-11, 1.269e-11,  
02341 1.209e-11, 1.162e-11, 1.129e-11, 1.108e-11, 1.099e-11, 1.103e-11,  
02342 1.119e-11, 1.148e-11, 1.193e-11, 1.252e-11, 1.329e-11, 1.421e-11,  
02343 1.555e-11, 1.685e-11, 1.839e-11, 2.054e-11, 2.317e-11, 2.571e-11,  
02344 2.839e-11, 3.171e-11, 3.49e-11, 3.886e-11, 4.287e-11, 4.645e-11,  
02345 5.047e-11, 5.592e-11, 6.109e-11, 6.628e-11, 7.381e-11, 8.088e-11,  
02346 8.966e-11, 1.045e-10, 1.12e-10, 1.287e-10, 1.486e-10, 1.662e-10,  
02347 1.866e-10, 2.133e-10, 2.524e-10, 2.776e-10, 3.204e-10, 3.559e-10,  
02348 4.028e-10, 4.448e-10, 4.882e-10, 5.244e-10, 5.605e-10, 6.018e-10,  
02349 6.328e-10, 6.579e-10, 6.541e-10, 7.024e-10, 7.074e-10, 7.068e-10,  
02350 7.009e-10, 6.698e-10, 6.545e-10, 6.209e-10, 5.834e-10, 5.412e-10,  
02351 5.001e-10, 4.231e-10, 3.727e-10, 3.211e-10, 2.833e-10, 2.447e-10,  
02352 2.097e-10, 1.843e-10, 1.639e-10, 1.449e-10, 1.27e-10, 1.161e-10,

```

02353 1.033e-10, 9.282e-11, 8.407e-11, 7.639e-11, 7.023e-11, 6.474e-11,
02354 6.142e-11, 5.76e-11, 5.568e-11, 5.472e-11, 5.39e-11, 5.455e-11,
02355 5.54e-11, 5.587e-11, 6.23e-11, 6.49e-11, 6.868e-11, 7.382e-11,
02356 8.022e-11, 8.372e-11, 9.243e-11, 1.004e-10, 1.062e-10, 1.13e-10,
02357 1.176e-10, 1.244e-10, 1.279e-10, 1.298e-10, 1.302e-10, 1.312e-10,
02358 1.295e-10, 1.244e-10, 1.211e-10, 1.167e-10, 1.098e-10, 9.927e-11,
02359 8.854e-11, 8.011e-11, 7.182e-11, 5.923e-11, 5.212e-11, 4.453e-11,
02360 3.832e-11, 3.371e-11, 2.987e-11, 2.651e-11, 2.354e-11, 2.093e-11,
02361 1.863e-11, 1.662e-11, 1.486e-11, 1.331e-11, 1.193e-11, 1.071e-11,
02362 9.628e-12, 8.66e-12, 7.801e-12, 7.031e-12, 6.347e-12, 5.733e-12,
02363 5.182e-12, 4.695e-12, 4.26e-12, 3.874e-12, 3.533e-12, 3.235e-12,
02364 2.979e-12, 2.76e-12, 2.579e-12, 2.432e-12, 2.321e-12, 2.246e-12,
02365 2.205e-12, 2.196e-12, 2.223e-12, 2.288e-12, 2.387e-12, 2.525e-12,
02366 2.704e-12, 2.925e-12, 3.191e-12, 3.508e-12, 3.876e-12, 4.303e-12,
02367 4.793e-12, 5.347e-12, 5.978e-12, 6.682e-12, 7.467e-12, 8.34e-12,
02368 9.293e-12, 1.035e-11, 1.152e-11, 1.285e-11, 1.428e-11, 1.586e-11,
02369 1.764e-11, 1.972e-11, 2.214e-11, 2.478e-11, 2.776e-11, 3.151e-11,
02370 3.591e-11, 4.103e-11, 4.66e-11, 5.395e-11, 6.306e-11, 7.172e-11,
02371 8.358e-11, 9.67e-11, 1.11e-10, 1.325e-10, 1.494e-10, 1.736e-10,
02372 2.007e-10, 2.296e-10, 2.608e-10, 3.004e-10, 3.361e-10, 3.727e-10,
02373 4.373e-10, 4.838e-10, 5.483e-10, 6.006e-10, 6.535e-10, 6.899e-10,
02374 7.687e-10, 8.444e-10, 8.798e-10, 9.135e-10, 9.532e-10, 9.757e-10,
02375 9.968e-10, 1.006e-9, 9.949e-10, 9.789e-10, 9.564e-10, 9.215e-10,
02376 8.51e-10, 8.394e-10, 7.707e-10, 7.152e-10, 6.274e-10, 5.598e-10,
02377 5.028e-10, 4.3e-10, 3.71e-10, 3.245e-10, 2.809e-10, 2.461e-10,
02378 2.154e-10, 1.91e-10, 1.685e-10, 1.487e-10, 1.313e-10, 1.163e-10,
02379 1.031e-10, 9.172e-11, 8.221e-11, 7.382e-11, 6.693e-11, 6.079e-11,
02380 5.581e-11, 5.167e-11, 4.811e-11, 4.506e-11, 4.255e-11, 4.083e-11,
02381 3.949e-11, 3.881e-11, 3.861e-11, 3.858e-11, 3.951e-11, 4.045e-11,
02382 4.24e-11, 4.487e-11, 4.806e-11, 5.133e-11, 5.518e-11, 5.919e-11,
02383 6.533e-11, 7.031e-11, 7.762e-11, 8.305e-11, 9.252e-11, 9.727e-11,
02384 1.045e-10, 1.117e-10, 1.2e-10, 1.275e-10, 1.341e-10, 1.362e-10,
02385 1.438e-10, 1.45e-10, 1.455e-10, 1.455e-10, 1.434e-10, 1.381e-10,
02386 1.301e-10, 1.276e-10, 1.163e-10, 1.089e-10, 9.911e-11, 8.943e-11,
02387 7.618e-11, 6.424e-11, 5.717e-11, 4.866e-11, 4.257e-11, 3.773e-11,
02388 3.331e-11, 2.958e-11, 2.629e-11, 2.316e-11, 2.073e-11, 1.841e-11,
02389 1.635e-11, 1.464e-11, 1.31e-11, 1.16e-11, 1.047e-11, 9.408e-12,
02390 8.414e-12, 7.521e-12, 6.705e-12, 5.993e-12, 5.371e-12, 4.815e-12,
02391 4.338e-12, 3.921e-12, 3.567e-12, 3.265e-12, 3.01e-12, 2.795e-12,
02392 2.613e-12, 2.464e-12, 2.346e-12, 2.256e-12, 2.195e-12, 2.165e-12,
02393 2.166e-12, 2.198e-12, 2.262e-12, 2.364e-12, 2.502e-12, 2.682e-12,
02394 2.908e-12, 3.187e-12, 3.533e-12, 3.946e-12, 4.418e-12, 5.013e-12,
02395 5.708e-12, 6.379e-12, 7.43e-12, 8.39e-12, 9.51e-12, 1.078e-11,
02396 1.259e-11, 1.438e-11, 1.63e-11, 1.814e-11, 2.055e-11, 2.348e-11,
02397 2.664e-11, 2.956e-11, 3.3e-11, 3.677e-11, 4.032e-11, 4.494e-11,
02398 4.951e-11, 5.452e-11, 6.014e-11, 6.5e-11, 6.915e-11, 7.45e-11,
02399 7.971e-11, 8.468e-11, 8.726e-11, 8.995e-11, 9.182e-11, 9.509e-11,
02400 9.333e-11, 9.386e-11, 9.457e-11, 9.21e-11, 9.019e-11, 8.68e-11,
02401 8.298e-11, 7.947e-11, 7.46e-11, 7.082e-11, 6.132e-11, 5.855e-11,
02402 5.073e-11, 4.464e-11, 3.825e-11, 3.375e-11, 2.911e-11, 2.535e-11,
02403 2.16e-11, 1.907e-11, 1.665e-11, 1.463e-11, 1.291e-11, 1.133e-11,
02404 9.997e-12, 8.836e-12, 7.839e-12, 6.943e-12, 6.254e-12, 5.6e-12,
02405 5.029e-12, 4.529e-12, 4.102e-12, 3.737e-12, 3.428e-12, 3.169e-12,
02406 2.959e-12, 2.798e-12, 2.675e-12, 2.582e-12, 2.644e-12, 2.557e-12,
02407 2.614e-12, 2.717e-12, 2.874e-12, 3.056e-12, 3.187e-12, 3.631e-12,
02408 3.979e-12, 4.248e-12, 4.817e-12, 5.266e-12, 5.836e-12, 6.365e-12,
02409 6.807e-12, 7.47e-12, 7.951e-12, 8.636e-12, 8.972e-12, 9.314e-12,
02410 9.445e-12, 1.003e-11, 1.013e-11, 9.937e-12, 9.729e-12, 9.064e-12,
02411 9.119e-12, 9.124e-12, 8.704e-12, 8.078e-12, 7.47e-12, 6.329e-12,
02412 5.674e-12, 4.808e-12, 4.119e-12, 3.554e-12, 3.103e-12, 2.731e-12,
02413 2.415e-12, 2.15e-12, 1.926e-12, 1.737e-12, 1.578e-12, 1.447e-12,
02414 1.34e-12, 1.255e-12, 1.191e-12, 1.146e-12, 1.121e-12, 1.114e-12,
02415 1.126e-12, 1.156e-12, 1.207e-12, 1.278e-12, 1.372e-12, 1.49e-12,
02416 1.633e-12, 1.805e-12, 2.01e-12, 2.249e-12, 2.528e-12, 2.852e-12,
02417 3.228e-12, 3.658e-12, 4.153e-12, 4.728e-12, 5.394e-12, 6.176e-12,
02418 7.126e-12, 8.188e-12, 9.328e-12, 1.103e-11, 1.276e-11, 1.417e-11,
02419 1.615e-11, 1.84e-11, 2.155e-11, 2.429e-11, 2.826e-11, 3.222e-11,
02420 3.664e-11, 4.14e-11, 4.906e-11, 5.536e-11, 6.327e-11, 7.088e-11,
02421 8.316e-11, 9.242e-11, 1.07e-10, 1.223e-10, 1.341e-10, 1.553e-10,
02422 1.703e-10, 1.9e-10, 2.022e-10, 2.233e-10, 2.345e-10, 2.438e-10,
02423 2.546e-10, 2.599e-10, 2.661e-10, 2.703e-10, 2.686e-10, 2.662e-10,
02424 2.56e-10, 2.552e-10, 2.378e-10, 2.252e-10, 2.146e-10, 1.885e-10,
02425 1.668e-10, 1.441e-10, 1.295e-10, 1.119e-10, 9.893e-11, 8.687e-11,
02426 7.678e-11, 6.685e-11, 5.879e-11, 5.127e-11, 4.505e-11, 3.997e-11,
02427 3.511e-11
02428 };
02429
02430 static double h2ofrn[2001] = { .01095, .01126, .01205, .01322, .0143,
02431 .01506, .01548, .01534, .01486, .01373, .01262, .01134, .01001,
02432 .008702, .007475, .006481, .00548, .0046, .003833, .00311,
02433 .002543, .002049, .00168, .001374, .001046, 8.193e-4, 6.267e-4,
02434 4.968e-4, 3.924e-4, 2.983e-4, 2.477e-4, 1.997e-4, 1.596e-4,
02435 1.331e-4, 1.061e-4, 8.942e-5, 7.168e-5, 5.887e-5, 4.848e-5,
02436 3.817e-5, 3.17e-5, 2.579e-5, 2.162e-5, 1.768e-5, 1.49e-5,
02437 1.231e-5, 1.013e-5, 8.555e-6, 7.328e-6, 6.148e-6, 5.207e-6,
02438 4.387e-6, 3.741e-6, 3.22e-6, 2.753e-6, 2.346e-6, 1.985e-6,
02439 1.716e-6, 1.475e-6, 1.286e-6, 1.122e-6, 9.661e-7, 8.284e-7,

```

02440 7.057e-7, 6.119e-7, 5.29e-7, 4.571e-7, 3.948e-7, 3.432e-7,  
02441 2.983e-7, 2.589e-7, 2.265e-7, 1.976e-7, 1.704e-7, 1.456e-7,  
02442 1.26e-7, 1.101e-7, 9.648e-8, 8.415e-8, 7.34e-8, 6.441e-8,  
02443 5.643e-8, 4.94e-8, 4.276e-8, 3.703e-8, 3.227e-8, 2.825e-8,  
02444 2.478e-8, 2.174e-8, 1.898e-8, 1.664e-8, 1.458e-8, 1.278e-8,  
02445 1.126e-8, 9.891e-9, 8.709e-9, 7.652e-9, 6.759e-9, 5.975e-9,  
02446 5.31e-9, 4.728e-9, 4.214e-9, 3.792e-9, 3.463e-9, 3.226e-9,  
02447 2.992e-9, 2.813e-9, 2.749e-9, 2.809e-9, 2.913e-9, 3.037e-9,  
02448 2.133e-9, 2.698e-9, 4.189e-9, 4.808e-9, 5.978e-9, 7.088e-9,  
02449 8.071e-9, 9.61e-9, 1.21e-8, 1.5e-8, 1.764e-8, 2.221e-8, 2.898e-8,  
02450 3.948e-8, 5.068e-8, 6.227e-8, 7.898e-8, 1.033e-7, 1.437e-7,  
02451 1.889e-7, 2.589e-7, 3.59e-7, 4.971e-7, 7.156e-7, 9.983e-7,  
02452 1.381e-6, 1.929e-6, 2.591e-6, 3.453e-6, 4.57e-6, 5.93e-6,  
02453 7.552e-6, 9.556e-6, 1.183e-5, 1.425e-5, 1.681e-5, 1.978e-5,  
02454 2.335e-5, 2.668e-5, 3.022e-5, 3.371e-5, 3.715e-5, 3.967e-5,  
02455 4.06e-5, 4.01e-5, 3.809e-5, 3.491e-5, 3.155e-5, 2.848e-5,  
02456 2.678e-5, 2.66e-5, 2.811e-5, 3.071e-5, 3.294e-5, 3.459e-5,  
02457 3.569e-5, 3.56e-5, 3.434e-5, 3.186e-5, 2.916e-5, 2.622e-5,  
02458 2.275e-5, 1.918e-5, 1.62e-5, 1.373e-5, 1.182e-5, 1.006e-5,  
02459 8.556e-6, 7.26e-6, 6.107e-6, 5.034e-6, 4.211e-6, 3.426e-6,  
02460 2.865e-6, 2.446e-6, 1.998e-6, 1.628e-6, 1.242e-6, 1.005e-6,  
02461 7.853e-7, 6.21e-7, 5.071e-7, 4.156e-7, 3.548e-7, 2.825e-7,  
02462 2.261e-7, 1.916e-7, 1.51e-7, 1.279e-7, 1.059e-7, 9.14e-8,  
02463 7.707e-8, 6.17e-8, 5.311e-8, 4.263e-8, 3.518e-8, 2.961e-8,  
02464 2.457e-8, 2.119e-8, 1.712e-8, 1.439e-8, 1.201e-8, 1.003e-8,  
02465 8.564e-9, 7.199e-9, 6.184e-9, 5.206e-9, 4.376e-9, 3.708e-9,  
02466 3.157e-9, 2.361e-9, 2.074e-9, 1.797e-9, 1.562e-9,  
02467 1.364e-9, 1.196e-9, 1.042e-9, 8.862e-10, 7.648e-10, 6.544e-10,  
02468 5.609e-10, 4.791e-10, 4.108e-10, 3.531e-10, 3.038e-10, 2.618e-10,  
02469 2.268e-10, 1.969e-10, 1.715e-10, 1.496e-10, 1.308e-10, 1.147e-10,  
02470 1.008e-10, 8.894e-11, 7.885e-11, 7.031e-11, 6.355e-11, 5.854e-11,  
02471 5.534e-11, 5.466e-11, 5.725e-11, 6.447e-11, 7.943e-11, 1.038e-10,  
02472 1.437e-10, 2.04e-10, 2.901e-10, 4.051e-10, 5.556e-10, 7.314e-10,  
02473 9.291e-10, 1.134e-9, 1.321e-9, 1.482e-9, 1.596e-9, 1.669e-9,  
02474 1.715e-9, 1.762e-9, 1.817e-9, 1.828e-9, 1.848e-9, 1.873e-9,  
02475 1.902e-9, 1.894e-9, 1.864e-9, 1.841e-9, 1.797e-9, 1.704e-9,  
02476 1.559e-9, 1.382e-9, 1.187e-9, 1.001e-9, 8.468e-10, 7.265e-10,  
02477 6.521e-10, 6.381e-10, 6.66e-10, 7.637e-10, 9.705e-10, 1.368e-9,  
02478 1.856e-9, 2.656e-9, 3.954e-9, 5.96e-9, 8.72e-9, 1.247e-8,  
02479 1.781e-8, 2.491e-8, 3.311e-8, 4.272e-8, 5.205e-8, 6.268e-8,  
02480 7.337e-8, 8.277e-8, 9.185e-8, 1.004e-7, 1.091e-7, 1.159e-7,  
02481 1.188e-7, 1.175e-7, 1.124e-7, 1.033e-7, 9.381e-8, 8.501e-8,  
02482 7.956e-8, 7.894e-8, 8.331e-8, 9.102e-8, 9.836e-8, 1.035e-7,  
02483 1.064e-7, 1.06e-7, 1.032e-7, 9.808e-8, 9.139e-8, 8.442e-8,  
02484 7.641e-8, 6.881e-8, 6.161e-8, 5.404e-8, 4.804e-8, 4.446e-8,  
02485 4.328e-8, 4.259e-8, 4.421e-8, 4.673e-8, 4.985e-8, 5.335e-8,  
02486 5.796e-8, 6.542e-8, 7.714e-8, 8.827e-8, 1.04e-7, 1.238e-7,  
02487 1.499e-7, 1.829e-7, 2.222e-7, 2.689e-7, 3.303e-7, 3.981e-7,  
02488 4.84e-7, 5.91e-7, 7.363e-7, 9.087e-7, 1.139e-6, 1.455e-6,  
02489 1.866e-6, 2.44e-6, 3.115e-6, 3.941e-6, 4.891e-6, 5.992e-6,  
02490 7.111e-6, 8.296e-6, 9.21e-6, 9.987e-6, 1.044e-5, 1.073e-5,  
02491 1.092e-5, 1.106e-5, 1.138e-5, 1.171e-5, 1.186e-5, 1.186e-5,  
02492 1.179e-5, 1.166e-5, 1.151e-5, 1.16e-5, 1.197e-5, 1.241e-5,  
02493 1.268e-5, 1.26e-5, 1.184e-5, 1.063e-5, 9.204e-6, 7.584e-6,  
02494 6.053e-6, 4.482e-6, 3.252e-6, 2.337e-6, 1.662e-6, 1.18e-6,  
02495 8.15e-7, 5.95e-7, 4.354e-7, 3.302e-7, 2.494e-7, 1.93e-7,  
02496 1.545e-7, 1.25e-7, 1.039e-7, 8.602e-8, 7.127e-8, 5.897e-8,  
02497 4.838e-8, 4.018e-8, 3.28e-8, 2.72e-8, 2.307e-8, 1.972e-8,  
02498 1.654e-8, 1.421e-8, 1.174e-8, 1.004e-8, 8.739e-9, 7.358e-9,  
02499 6.242e-9, 5.303e-9, 4.567e-9, 3.94e-9, 3.375e-9, 2.864e-9,  
02500 2.422e-9, 2.057e-9, 1.75e-9, 1.505e-9, 1.294e-9, 1.101e-9,  
02501 9.401e-10, 8.018e-10, 6.903e-10, 5.965e-10, 5.087e-10, 4.364e-10,  
02502 3.759e-10, 3.247e-10, 2.809e-10, 2.438e-10, 2.123e-10, 1.853e-10,  
02503 1.622e-10, 1.426e-10, 1.26e-10, 1.125e-10, 1.022e-10, 9.582e-11,  
02504 9.388e-11, 9.801e-11, 1.08e-10, 1.276e-10, 1.551e-10, 1.903e-10,  
02505 2.291e-10, 2.724e-10, 3.117e-10, 3.4e-10, 3.562e-10, 3.625e-10,  
02506 3.619e-10, 3.429e-10, 3.221e-10, 2.943e-10, 2.645e-10, 2.338e-10,  
02507 2.062e-10, 1.901e-10, 1.814e-10, 1.827e-10, 1.906e-10, 1.984e-10,  
02508 2.04e-10, 2.068e-10, 2.075e-10, 2.018e-10, 1.959e-10, 1.897e-10,  
02509 1.852e-10, 1.791e-10, 1.696e-10, 1.634e-10, 1.598e-10, 1.561e-10,  
02510 1.518e-10, 1.443e-10, 1.377e-10, 1.346e-10, 1.342e-10, 1.375e-10,  
02511 1.525e-10, 1.767e-10, 2.108e-10, 2.524e-10, 2.981e-10, 3.477e-10,  
02512 4.262e-10, 5.326e-10, 6.646e-10, 8.321e-10, 1.069e-9, 1.386e-9,  
02513 1.743e-9, 2.216e-9, 2.808e-9, 3.585e-9, 4.552e-9, 5.907e-9,  
02514 7.611e-9, 7.774e-9, 1.255e-8, 1.666e-8, 2.279e-8, 3.221e-8,  
02515 4.531e-8, 6.4e-8, 9.187e-8, 1.295e-7, 1.825e-7, 2.431e-7,  
02516 3.181e-7, 4.009e-7, 4.941e-7, 5.88e-7, 6.623e-7, 7.155e-7,  
02517 7.451e-7, 7.594e-7, 7.541e-7, 7.467e-7, 7.527e-7, 7.935e-7,  
02518 8.461e-7, 8.954e-7, 9.364e-7, 9.843e-7, 1.024e-6, 1.05e-6,  
02519 1.059e-6, 1.074e-6, 1.072e-6, 1.043e-6, 9.789e-7, 8.803e-7,  
02520 7.662e-7, 6.378e-7, 5.133e-7, 3.958e-7, 2.914e-7, 2.144e-7,  
02521 1.57e-7, 1.14e-7, 8.47e-8, 6.2e-8, 4.657e-8, 3.559e-8, 2.813e-8,  
02522 2.222e-8, 1.769e-8, 1.391e-8, 1.125e-8, 9.186e-9, 7.704e-9,  
02523 6.447e-9, 5.381e-9, 4.442e-9, 3.669e-9, 3.057e-9, 2.564e-9,  
02524 2.153e-9, 1.784e-9, 1.499e-9, 1.281e-9, 1.082e-9, 9.304e-10,  
02525 8.169e-10, 6.856e-10, 5.866e-10, 5.043e-10, 4.336e-10, 3.731e-10,  
02526 3.175e-10, 2.745e-10, 2.374e-10, 2.007e-10, 1.737e-10, 1.508e-10,

```

02527 1.302e-10, 1.13e-10, 9.672e-11, 8.375e-11, 7.265e-11, 6.244e-11,
02528 5.343e-11, 4.654e-11, 3.975e-11, 3.488e-11, 3.097e-11, 2.834e-11,
02529 2.649e-11, 2.519e-11, 2.462e-11, 2.443e-11, 2.44e-11, 2.398e-11,
02530 2.306e-11, 2.183e-11, 2.021e-11, 1.821e-11, 1.599e-11, 1.403e-11,
02531 1.196e-11, 1.023e-11, 8.728e-12, 7.606e-12, 6.941e-12, 6.545e-12,
02532 6.484e-12, 6.6e-12, 6.718e-12, 6.785e-12, 6.746e-12, 6.724e-12,
02533 6.764e-12, 6.995e-12, 7.144e-12, 7.32e-12, 7.33e-12, 7.208e-12,
02534 6.789e-12, 6.09e-12, 5.337e-12, 4.62e-12, 4.037e-12, 3.574e-12,
02535 3.311e-12, 3.546e-12, 3.566e-12, 3.836e-12, 4.076e-12, 4.351e-12,
02536 4.691e-12, 5.114e-12, 5.427e-12, 6.167e-12, 7.436e-12, 8.842e-12,
02537 1.038e-11, 1.249e-11, 1.54e-11, 1.915e-11, 2.48e-11, 3.256e-11,
02538 4.339e-11, 5.611e-11, 7.519e-11, 1.037e-10, 1.409e-10, 1.883e-10,
02539 2.503e-10, 3.38e-10, 4.468e-10, 5.801e-10, 7.335e-10, 8.98e-10,
02540 1.11e-9, 1.363e-9, 1.677e-9, 2.104e-9, 2.681e-9, 3.531e-9,
02541 4.621e-9, 6.106e-9, 8.154e-9, 1.046e-8, 1.312e-8, 1.607e-8,
02542 1.948e-8, 2.266e-8, 2.495e-8, 2.655e-8, 2.739e-8, 2.739e-8,
02543 2.662e-8, 2.589e-8, 2.59e-8, 2.664e-8, 2.833e-8, 3.023e-8,
02544 3.305e-8, 3.558e-8, 3.793e-8, 3.961e-8, 4.056e-8, 4.102e-8,
02545 4.025e-8, 3.917e-8, 3.706e-8, 3.493e-8, 3.249e-8, 3.096e-8,
02546 3.011e-8, 3.111e-8, 3.395e-8, 3.958e-8, 4.875e-8, 6.066e-8,
02547 7.915e-8, 1.011e-7, 1.3e-7, 1.622e-7, 2.003e-7, 2.448e-7,
02548 2.863e-7, 3.317e-7, 3.655e-7, 3.96e-7, 4.098e-7, 4.168e-7,
02549 4.198e-7, 4.207e-7, 4.289e-7, 4.384e-7, 4.471e-7, 4.524e-7,
02550 4.574e-7, 4.633e-7, 4.785e-7, 5.028e-7, 5.371e-7, 5.727e-7,
02551 5.955e-7, 5.998e-7, 5.669e-7, 5.082e-7, 4.397e-7, 3.596e-7,
02552 2.814e-7, 2.074e-7, 1.486e-7, 1.057e-7, 7.25e-8, 4.946e-8,
02553 3.43e-8, 2.447e-8, 1.793e-8, 1.375e-8, 1.096e-8, 9.091e-9,
02554 7.709e-9, 6.631e-9, 5.714e-9, 4.886e-9, 4.205e-9, 3.575e-9,
02555 3.07e-9, 2.631e-9, 2.284e-9, 2.002e-9, 1.745e-9, 1.509e-9,
02556 1.284e-9, 1.084e-9, 9.163e-10, 7.663e-10, 6.346e-10, 5.283e-10,
02557 4.354e-10, 3.59e-10, 2.982e-10, 2.455e-10, 2.033e-10, 1.696e-10,
02558 1.432e-10, 1.211e-10, 1.02e-10, 8.702e-11, 7.38e-11, 6.293e-11,
02559 5.343e-11, 4.532e-11, 3.907e-11, 3.365e-11, 2.945e-11, 2.558e-11,
02560 2.192e-11, 1.895e-11, 1.636e-11, 1.42e-11, 1.228e-11, 1.063e-11,
02561 9.348e-12, 8.2e-12, 7.231e-12, 6.43e-12, 5.702e-12, 5.052e-12,
02562 4.469e-12, 4e-12, 3.679e-12, 3.387e-12, 3.197e-12, 3.158e-12,
02563 3.327e-12, 3.675e-12, 4.292e-12, 5.437e-12, 7.197e-12, 1.008e-11,
02564 1.437e-11, 2.035e-11, 2.905e-11, 4.062e-11, 5.528e-11, 7.177e-11,
02565 9.064e-11, 1.109e-10, 1.297e-10, 1.473e-10, 1.652e-10, 1.851e-10,
02566 2.079e-10, 2.313e-10, 2.619e-10, 2.958e-10, 3.352e-10, 3.796e-10,
02567 4.295e-10, 4.923e-10, 5.49e-10, 5.998e-10, 6.388e-10, 6.645e-10,
02568 6.712e-10, 6.549e-10, 6.38e-10, 6.255e-10, 6.253e-10, 6.459e-10,
02569 6.977e-10, 7.59e-10, 8.242e-10, 8.92e-10, 9.403e-10, 9.701e-10,
02570 9.483e-10, 9.135e-10, 8.617e-10, 7.921e-10, 7.168e-10, 6.382e-10,
02571 5.677e-10, 5.045e-10, 4.572e-10, 4.312e-10, 4.145e-10, 4.192e-10,
02572 4.541e-10, 5.368e-10, 6.771e-10, 8.962e-10, 1.21e-9, 1.659e-9,
02573 2.33e-9, 3.249e-9, 4.495e-9, 5.923e-9, 7.642e-9, 9.607e-9,
02574 1.178e-8, 1.399e-8, 1.584e-8, 1.73e-8, 1.816e-8, 1.87e-8,
02575 1.868e-8, 1.87e-8, 1.884e-8, 1.99e-8, 2.15e-8, 2.258e-8,
02576 2.364e-8, 2.473e-8, 2.602e-8, 2.689e-8, 2.731e-8, 2.816e-8,
02577 2.859e-8, 2.839e-8, 2.703e-8, 2.451e-8, 2.149e-8, 1.787e-8,
02578 1.449e-8, 1.111e-8, 8.282e-9, 6.121e-9, 4.494e-9, 3.367e-9,
02579 2.487e-9, 1.885e-9, 1.503e-9, 1.249e-9, 1.074e-9, 9.427e-10,
02580 8.439e-10, 7.563e-10, 6.772e-10, 6.002e-10, 5.254e-10, 4.588e-10,
02581 3.977e-10, 3.449e-10, 3.003e-10, 2.624e-10, 2.335e-10, 2.04e-10,
02582 1.771e-10, 1.534e-10, 1.296e-10, 1.097e-10, 9.173e-11, 7.73e-11,
02583 6.547e-11, 5.191e-11, 4.198e-11, 3.361e-11, 2.732e-11, 2.244e-11,
02584 1.791e-11, 1.509e-11, 1.243e-11, 1.035e-11, 8.969e-12, 7.394e-12,
02585 6.323e-12, 5.282e-12, 4.543e-12, 3.752e-12, 3.14e-12, 2.6e-12,
02586 2.194e-12, 1.825e-12, 1.511e-12, 1.245e-12, 1.024e-12, 8.539e-13,
02587 7.227e-13, 6.102e-13, 5.189e-13, 4.43e-13, 3.774e-13, 3.236e-13,
02588 2.8e-13, 2.444e-13, 2.156e-13, 1.932e-13, 1.775e-13, 1.695e-13,
02589 1.672e-13, 1.704e-13, 1.825e-13, 2.087e-13, 2.614e-13, 3.377e-13,
02590 4.817e-13, 6.989e-13, 1.062e-12, 1.562e-12, 2.288e-12, 3.295e-12,
02591 4.55e-12, 5.965e-12, 7.546e-12, 9.395e-12, 1.103e-11, 1.228e-11,
02592 1.318e-11, 1.38e-11, 1.421e-11, 1.39e-11, 1.358e-11, 1.336e-11,
02593 1.342e-11, 1.356e-11, 1.424e-11, 1.552e-11, 1.73e-11, 1.951e-11,
02594 2.128e-11, 2.249e-11, 2.277e-11, 2.226e-11, 2.111e-11, 1.922e-11,
02595 1.775e-11, 1.661e-11, 1.547e-11, 1.446e-11, 1.323e-11, 1.21e-11,
02596 1.054e-11, 9.283e-12, 8.671e-12, 8.67e-12, 9.429e-12, 1.062e-11,
02597 1.255e-11, 1.506e-11, 1.818e-11, 2.26e-11, 2.831e-11, 3.723e-11,
02598 5.092e-11, 6.968e-11, 9.826e-11, 1.349e-10, 1.87e-10, 2.58e-10,
02599 3.43e-10, 4.424e-10, 5.521e-10, 6.812e-10, 8.064e-10, 9.109e-10,
02600 9.839e-10, 1.028e-9, 1.044e-9, 1.029e-9, 1.005e-9, 1.002e-9,
02601 1.038e-9, 1.122e-9, 1.233e-9, 1.372e-9, 1.524e-9, 1.665e-9,
02602 1.804e-9, 1.908e-9, 2.015e-9, 2.117e-9, 2.219e-9, 2.336e-9,
02603 2.531e-9, 2.805e-9, 3.189e-9, 3.617e-9, 4.208e-9, 4.911e-9,
02604 5.619e-9, 6.469e-9, 7.188e-9, 7.957e-9, 8.503e-9, 9.028e-9,
02605 9.571e-9, 9.99e-9, 1.055e-8, 1.102e-8, 1.132e-8, 1.141e-8,
02606 1.145e-8, 1.145e-8, 1.176e-8, 1.224e-8, 1.304e-8, 1.388e-8,
02607 1.445e-8, 1.453e-8, 1.368e-8, 1.22e-8, 1.042e-8, 8.404e-9,
02608 6.403e-9, 6.443e-9, 3.325e-9, 2.335e-9, 1.638e-9, 1.19e-9,
02609 9.161e-10, 7.412e-10, 6.226e-10, 5.516e-10, 5.068e-10, 4.831e-10,
02610 4.856e-10, 5.162e-10, 5.785e-10, 6.539e-10, 7.485e-10, 8.565e-10,
02611 9.534e-10, 1.052e-9, 1.115e-9, 1.173e-9, 1.203e-9, 1.224e-9,
02612 1.243e-9, 1.248e-9, 1.261e-9, 1.265e-9, 1.25e-9, 1.217e-9,
02613 1.176e-9, 1.145e-9, 1.153e-9, 1.199e-9, 1.278e-9, 1.366e-9,

```

02614 1.426e-9, 1.444e-9, 1.365e-9, 1.224e-9, 1.051e-9, 8.539e-10,  
02615 6.564e-10, 4.751e-10, 3.404e-10, 2.377e-10, 1.631e-10, 1.114e-10,  
02616 7.87e-11, 5.793e-11, 4.284e-11, 3.3e-11, 2.62e-11, 2.152e-11,  
02617 1.777e-11, 1.496e-11, 1.242e-11, 1.037e-11, 8.725e-12, 7.004e-12,  
02618 5.718e-12, 4.769e-12, 3.952e-12, 3.336e-12, 2.712e-12, 2.213e-12,  
02619 1.803e-12, 1.492e-12, 1.236e-12, 1.006e-12, 8.384e-13, 7.063e-13,  
02620 5.879e-13, 4.93e-13, 4.171e-13, 3.569e-13, 3.083e-13, 2.688e-13,  
02621 2.333e-13, 2.035e-13, 1.82e-13, 1.682e-13, 1.635e-13, 1.674e-13,  
02622 1.769e-13, 2.022e-13, 2.485e-13, 3.127e-13, 4.25e-13, 5.928e-13,  
02623 8.514e-13, 1.236e-12, 1.701e-12, 2.392e-12, 3.231e-12, 4.35e-12,  
02624 5.559e-12, 6.915e-12, 8.519e-12, 1.013e-11, 1.146e-11, 1.24e-11,  
02625 1.305e-11, 1.333e-11, 1.318e-11, 1.263e-11, 1.238e-11, 1.244e-11,  
02626 1.305e-11, 1.432e-11, 1.623e-11, 1.846e-11, 2.09e-11, 2.328e-11,  
02627 2.526e-11, 2.637e-11, 2.702e-11, 2.794e-11, 2.889e-11, 2.989e-11,  
02628 3.231e-11, 3.68e-11, 4.375e-11, 5.504e-11, 7.159e-11, 9.502e-11,  
02629 1.279e-10, 1.645e-10, 2.098e-10, 2.618e-10, 3.189e-10, 3.79e-10,  
02630 4.303e-10, 4.753e-10, 5.027e-10, 5.221e-10, 5.293e-10, 5.346e-10,  
02631 5.467e-10, 5.796e-10, 6.2e-10, 6.454e-10, 6.705e-10, 6.925e-10,  
02632 7.233e-10, 7.35e-10, 7.538e-10, 7.861e-10, 8.077e-10, 8.132e-10,  
02633 7.749e-10, 7.036e-10, 6.143e-10, 5.093e-10, 4.089e-10, 3.092e-10,  
02634 2.299e-10, 1.705e-10, 1.277e-10, 9.723e-11, 7.533e-11, 6.126e-11,  
02635 5.154e-11, 4.428e-11, 3.913e-11, 3.521e-11, 3.297e-11, 3.275e-11,  
02636 3.46e-11, 3.798e-11, 4.251e-11, 4.745e-11, 5.232e-11, 5.606e-11,  
02637 5.82e-11, 5.88e-11, 5.79e-11, 5.661e-11, 5.491e-11, 5.366e-11,  
02638 5.341e-11, 5.353e-11, 5.336e-11, 5.293e-11, 5.248e-11, 5.235e-11,  
02639 5.208e-11, 5.322e-11, 5.521e-11, 5.725e-11, 5.827e-11, 5.685e-11,  
02640 5.245e-11, 4.612e-11, 3.884e-11, 3.129e-11, 2.404e-11, 1.732e-11,  
02641 1.223e-11, 8.574e-12, 5.888e-12, 3.986e-12, 2.732e-12, 1.948e-12,  
02642 1.414e-12, 1.061e-12, 8.298e-13, 6.612e-13, 5.413e-13, 4.472e-13,  
02643 3.772e-13, 3.181e-13, 2.645e-13, 2.171e-13, 1.778e-13, 1.464e-13,  
02644 1.183e-13, 9.637e-14, 7.991e-14, 6.668e-14, 5.57e-14, 4.663e-14,  
02645 3.848e-14, 3.233e-14, 2.706e-14, 2.284e-14, 1.944e-14, 1.664e-14,  
02646 1.43e-14, 1.233e-14, 1.066e-14, 9.234e-15, 8.023e-15, 6.993e-15,  
02647 6.119e-15, 5.384e-15, 4.774e-15, 4.283e-15, 3.916e-15, 3.695e-15,  
02648 3.682e-15, 4.004e-15, 4.912e-15, 6.853e-15, 1.056e-14, 1.712e-14,  
02649 2.804e-14, 4.516e-14, 7.113e-14, 1.084e-13, 1.426e-13, 1.734e-13,  
02650 1.978e-13, 2.194e-13, 2.388e-13, 2.489e-13, 2.626e-13, 2.865e-13,  
02651 3.105e-13, 3.387e-13, 3.652e-13, 3.984e-13, 4.398e-13, 4.906e-13,  
02652 5.55e-13, 6.517e-13, 7.813e-13, 9.272e-13, 1.164e-12, 1.434e-12,  
02653 1.849e-12, 2.524e-12, 3.328e-12, 4.523e-12, 6.108e-12, 8.207e-12,  
02654 1.122e-11, 1.477e-11, 1.9e-11, 2.412e-11, 2.984e-11, 3.68e-11,  
02655 4.353e-11, 4.963e-11, 5.478e-11, 5.903e-11, 6.233e-11, 6.483e-11,  
02656 6.904e-11, 7.569e-11, 8.719e-11, 1.048e-10, 1.278e-10, 1.557e-10,  
02657 1.869e-10, 2.218e-10, 2.61e-10, 2.975e-10, 3.371e-10, 3.746e-10,  
02658 4.065e-10, 4.336e-10, 4.503e-10, 4.701e-10, 4.8e-10, 4.917e-10,  
02659 5.038e-10, 5.128e-10, 5.143e-10, 5.071e-10, 5.019e-10, 5.025e-10,  
02660 5.183e-10, 5.496e-10, 5.877e-10, 6.235e-10, 6.42e-10, 6.234e-10,  
02661 5.698e-10, 4.916e-10, 4.022e-10, 3.126e-10, 2.282e-10, 1.639e-10,  
02662 1.142e-10, 7.919e-11, 5.69e-11, 4.313e-11, 3.413e-11, 2.807e-11,  
02663 2.41e-11, 2.166e-11, 2.024e-11, 1.946e-11, 1.929e-11, 1.963e-11,  
02664 2.035e-11, 2.162e-11, 2.305e-11, 2.493e-11, 2.748e-11, 3.048e-11,  
02665 3.413e-11, 3.754e-11, 4.155e-11, 4.635e-11, 5.11e-11, 5.734e-11,  
02666 6.338e-11, 6.99e-11, 7.611e-11, 8.125e-11, 8.654e-11, 8.951e-11,  
02667 9.182e-11, 9.31e-11, 9.273e-11, 9.094e-11, 8.849e-11, 8.662e-11,  
02668 8.67e-11, 8.972e-11, 9.566e-11, 1.025e-10, 1.083e-10, 1.111e-10,  
02669 1.074e-10, 9.771e-11, 8.468e-11, 6.958e-11, 5.47e-11, 4.04e-11,  
02670 2.94e-11, 2.075e-11, 1.442e-11, 1.01e-11, 7.281e-12, 5.409e-12,  
02671 4.138e-12, 3.304e-12, 2.784e-12, 2.473e-12, 2.273e-12, 2.186e-12,  
02672 2.118e-12, 2.066e-12, 1.958e-12, 1.818e-12, 1.675e-12, 1.509e-12,  
02673 1.349e-12, 1.171e-12, 9.838e-13, 8.213e-13, 6.765e-13, 5.378e-13,  
02674 4.161e-13, 3.119e-13, 2.279e-13, 1.637e-13, 1.152e-13, 8.112e-14,  
02675 5.919e-14, 4.47e-14, 3.492e-14, 2.811e-14, 2.319e-14, 1.948e-14,  
02676 1.66e-14, 1.432e-14, 1.251e-14, 1.109e-14, 1.006e-14, 9.45e-15,  
02677 9.384e-15, 1.012e-14, 1.216e-14, 1.636e-14, 2.305e-14, 3.488e-14,  
02678 5.572e-14, 8.479e-14, 1.265e-13, 1.905e-13, 2.73e-13, 3.809e-13,  
02679 4.955e-13, 6.303e-13, 7.861e-13, 9.427e-13, 1.097e-12, 1.212e-12,  
02680 1.328e-12, 1.415e-12, 1.463e-12, 1.495e-12, 1.571e-12, 1.731e-12,  
02681 1.981e-12, 2.387e-12, 2.93e-12, 3.642e-12, 4.584e-12, 5.822e-12,  
02682 7.278e-12, 9.193e-12, 1.135e-11, 1.382e-11, 1.662e-11, 1.958e-11,  
02683 2.286e-11, 2.559e-11, 2.805e-11, 2.988e-11, 3.106e-11, 3.182e-11,  
02684 3.2e-11, 3.258e-11, 3.362e-11, 3.558e-11, 3.688e-11, 3.8e-11,  
02685 3.929e-11, 4.062e-11, 4.186e-11, 4.293e-11, 4.48e-11, 4.643e-11,  
02686 4.704e-11, 4.571e-11, 4.206e-11, 3.715e-11, 3.131e-11, 2.541e-11,  
02687 1.978e-11, 1.508e-11, 1.146e-11, 8.7e-12, 6.603e-12, 5.162e-12,  
02688 4.157e-12, 3.408e-12, 2.829e-12, 2.405e-12, 2.071e-12, 1.826e-12,  
02689 1.648e-12, 1.542e-12, 1.489e-12, 1.485e-12, 1.493e-12, 1.545e-12,  
02690 1.637e-12, 1.814e-12, 2.061e-12, 2.312e-12, 2.651e-12, 3.03e-12,  
02691 3.46e-12, 3.901e-12, 4.306e-12, 4.721e-12, 5.008e-12, 5.281e-12,  
02692 5.541e-12, 5.791e-12, 6.115e-12, 6.442e-12, 6.68e-12, 6.791e-12,  
02693 6.831e-12, 6.839e-12, 6.946e-12, 7.128e-12, 7.537e-12, 8.036e-12,  
02694 8.392e-12, 8.526e-12, 8.11e-12, 7.325e-12, 6.329e-12, 5.183e-12,  
02695 4.081e-12, 2.985e-12, 2.141e-12, 1.492e-12, 1.015e-12, 6.684e-13,  
02696 4.414e-13, 2.987e-13, 2.038e-13, 1.391e-13, 9.86e-14, 7.24e-14,  
02697 5.493e-14, 4.288e-14, 3.427e-14, 2.787e-14, 2.296e-14, 1.909e-14,  
02698 1.598e-14, 1.344e-14, 1.135e-14, 9.616e-15, 8.169e-15, 6.957e-15,  
02699 5.938e-15, 5.08e-15, 4.353e-15, 3.738e-15, 3.217e-15, 2.773e-15,  
02700 2.397e-15, 2.077e-15, 1.805e-15, 1.575e-15, 1.382e-15, 1.221e-15,

```

02701    1.09e-15, 9.855e-16, 9.068e-16, 8.537e-16, 8.27e-16, 8.29e-16,
02702    8.634e-16, 9.359e-16, 1.055e-15, 1.233e-15, 1.486e-15, 1.839e-15,
02703    2.326e-15, 2.998e-15, 3.934e-15, 5.256e-15, 7.164e-15, 9.984e-15,
02704    1.427e-14, 2.099e-14, 3.196e-14, 5.121e-14, 7.908e-14, 1.131e-13,
02705    1.602e-13, 2.239e-13, 3.075e-13, 4.134e-13, 5.749e-13, 7.886e-13,
02706    1.071e-12, 1.464e-12, 2.032e-12, 2.8e-12, 3.732e-12, 4.996e-12,
02707    6.483e-12, 8.143e-12, 1.006e-11, 1.238e-11, 1.484e-11, 1.744e-11,
02708    2.02e-11, 2.274e-11, 2.562e-11, 2.848e-11, 3.191e-11, 3.617e-11,
02709    4.081e-11, 4.577e-11, 4.937e-11, 5.204e-11, 5.401e-11, 5.462e-11,
02710    5.507e-11, 5.51e-11, 5.605e-11, 5.686e-11, 5.739e-11, 5.766e-11,
02711    5.74e-11, 5.754e-11, 5.761e-11, 5.777e-11, 5.712e-11, 5.51e-11,
02712    5.088e-11, 4.438e-11, 3.728e-11, 2.994e-11, 2.305e-11, 1.715e-11,
02713    1.256e-11, 9.208e-12, 6.745e-12, 5.014e-12, 3.785e-12, 2.9e-12,
02714    2.239e-12, 1.757e-12, 1.414e-12, 1.142e-12, 9.482e-13, 8.01e-13,
02715    6.961e-13, 6.253e-13, 5.735e-13, 5.433e-13, 5.352e-13, 5.493e-13,
02716    5.706e-13, 6.068e-13, 6.531e-13, 7.109e-13, 7.767e-13, 8.59e-13,
02717    9.792e-13, 1.142e-12, 1.371e-12, 1.65e-12, 1.957e-12, 2.302e-12,
02718    2.705e-12, 3.145e-12, 3.608e-12, 4.071e-12, 4.602e-12, 5.133e-12,
02719    5.572e-12, 5.987e-12, 6.248e-12, 6.533e-12, 6.757e-12, 6.935e-12,
02720    7.224e-12, 7.422e-12, 7.538e-12, 7.547e-12, 7.495e-12, 7.543e-12,
02721    7.725e-12, 8.139e-12, 8.627e-12, 9.146e-12, 9.443e-12, 9.318e-12,
02722    8.649e-12, 7.512e-12, 6.261e-12, 4.915e-12, 3.647e-12, 2.597e-12,
02723    1.785e-12, 1.242e-12, 8.66e-13, 6.207e-13, 4.61e-13, 3.444e-13,
02724    2.634e-13, 2.1e-13, 1.725e-13, 1.455e-13, 1.237e-13, 1.085e-13,
02725    9.513e-14, 7.978e-14, 6.603e-14, 5.288e-14, 4.084e-14, 2.952e-14,
02726    2.157e-14, 1.593e-14, 1.199e-14, 9.267e-15, 7.365e-15, 6.004e-15,
02727    4.995e-15, 4.218e-15, 3.601e-15, 3.101e-15, 2.692e-15, 2.36e-15,
02728    2.094e-15, 1.891e-15, 1.755e-15, 1.699e-15, 1.755e-15, 1.987e-15,
02729    2.506e-15, 3.506e-15, 5.289e-15, 8.311e-15, 1.325e-14, 2.129e-14,
02730    3.237e-14, 4.595e-14, 6.441e-14, 8.433e-14, 1.074e-13, 1.383e-13,
02731    1.762e-13, 2.281e-13, 2.831e-13, 3.523e-13, 4.38e-13, 5.304e-13,
02732    6.29e-13, 7.142e-13, 8.032e-13, 8.934e-13, 9.888e-13, 1.109e-12,
02733    1.261e-12, 1.462e-12, 1.74e-12, 2.099e-12, 2.535e-12, 3.008e-12,
02734    3.462e-12, 3.856e-12, 4.098e-12, 4.239e-12, 4.234e-12, 4.132e-12,
02735    3.986e-12, 3.866e-12, 3.829e-12, 3.742e-12, 3.705e-12, 3.694e-12,
02736    3.765e-12, 3.849e-12, 3.929e-12, 4.056e-12, 4.092e-12, 4.047e-12,
02737    3.792e-12, 3.407e-12, 2.953e-12, 2.429e-12, 1.931e-12, 1.46e-12,
02738    1.099e-12, 8.199e-13, 6.077e-13, 4.449e-13, 3.359e-13, 2.524e-13,
02739    1.881e-13, 1.391e-13, 1.02e-13, 7.544e-14, 5.555e-14, 4.22e-14,
02740    3.321e-14, 2.686e-14, 2.212e-14, 1.78e-14, 1.369e-14, 1.094e-14,
02741    9.13e-15, 8.101e-15, 7.828e-15, 8.393e-15, 1.012e-14, 1.259e-14,
02742    1.538e-14, 1.961e-14, 2.619e-14, 3.679e-14, 5.049e-14, 6.917e-14,
02743    8.88e-14, 1.115e-13, 1.373e-13, 1.619e-13, 1.878e-13, 2.111e-13,
02744    2.33e-13, 2.503e-13, 2.613e-13, 2.743e-13, 2.826e-13, 2.976e-13,
02745    3.162e-13, 3.36e-13, 3.491e-13, 3.541e-13, 3.595e-13, 3.608e-13,
02746    3.709e-13, 3.869e-13, 4.12e-13, 4.366e-13, 4.504e-13, 4.379e-13,
02747    3.955e-13, 3.385e-13, 2.741e-13, 2.089e-13, 1.427e-13, 9.294e-14,
02748    5.775e-14, 3.565e-14, 2.21e-14, 1.398e-14, 9.194e-15, 6.363e-15,
02749    4.644e-15, 3.55e-15, 2.808e-15, 2.274e-15, 1.871e-15, 1.557e-15,
02750    1.308e-15, 1.108e-15, 9.488e-16, 8.222e-16, 7.238e-16, 6.506e-16,
02751    6.008e-16, 5.742e-16, 5.724e-16, 5.991e-16, 6.625e-16, 7.775e-16,
02752    9.734e-16, 1.306e-15, 1.88e-15, 2.879e-15, 4.616e-15, 7.579e-15,
02753    1.248e-14, 2.03e-14, 3.244e-14, 5.171e-14, 7.394e-14, 9.676e-14,
02754    1.199e-13, 1.467e-13, 1.737e-13, 2.02e-13, 2.425e-13, 3.016e-13,
02755    3.7e-13, 4.617e-13, 5.949e-13, 7.473e-13, 9.378e-13, 1.191e-12,
02756    1.481e-12, 1.813e-12, 2.232e-12, 2.722e-12, 3.254e-12, 3.845e-12,
02757    4.458e-12, 5.048e-12, 5.511e-12, 5.898e-12, 6.204e-12, 6.293e-12,
02758    6.386e-12, 6.467e-12, 6.507e-12, 6.466e-12, 6.443e-12, 6.598e-12,
02759    6.873e-12, 7.3e-12, 7.816e-12, 8.368e-12, 8.643e-12, 8.466e-12,
02760    7.871e-12, 6.853e-12, 5.714e-12, 4.482e-12, 3.392e-12, 2.613e-12,
02761    2.008e-12, 1.562e-12, 1.228e-12, 9.888e-13, 7.646e-13, 5.769e-13,
02762    4.368e-13, 3.324e-13, 2.508e-13, 1.916e-13
02763 };
02764
02765 static double xfcrev[15] =
02766 { 1.003, 1.009, 1.015, 1.023, 1.029, 1.033, 1.037,
02767   1.039, 1.04, 1.046, 1.036, 1.027, 1.01, 1.002, 1.
02768 };
02769
02770 double sfac;
02771
02772 /* Get H2O continuum absorption... */
02773 double xw = nu / 10 + 1;
02774 if (xw >= 1 && xw < 2001) {
02775     int iw = (int) xw;
02776     double dw = xw - iw;
02777     double ew = 1 - dw;
02778     double cw296 = ew * h2o296[iw - 1] + dw * h2o296[iw];
02779     double cw260 = ew * h2o260[iw - 1] + dw * h2o260[iw];
02780     double cwfrn = ew * h2ofrn[iw - 1] + dw * h2ofrn[iw];
02781     if (nu <= 820 || nu >= 960) {
02782         sfac = 1;
02783     } else {
02784         double xx = (nu - 820) / 10;
02785         int ix = (int) xx;
02786         double dx = xx - ix;
02787         sfac = (1 - dx) * xfcrev[ix] + dx * xfcrev[ix + 1];

```



```

02788     }
02789     double ctws1f =
02790         sfac * cw296 * pow(cw260 / cw296, (296 - t) / (296 - 260));
02791     double vf2 = POW2(nu - 370);
02792     double vf6 = POW3(vf2);
02793     double fscal = 36100 / (vf2 + vf6 * 1e-8 + 36100) * -.25 + 1;
02794     double ctwfrn = cwfrn * fscal;
02795     double a1 = nu * u * tanh(.7193876 / t * nu);
02796     double a2 = 296 / t;
02797     double a3 = p / P0 * (q * ctws1f + (1 - q) * ctwfrn) * 1e-20;
02798     return a1 * a2 * a3;
02799 } else
02800     return 0;
02801 }
02802
02803 /*****
02804
02805 double ctmn2(
02806     double nu,
02807     double p,
02808     double t) {
02809
02810     static double ba[98] = { 0., 4.45e-8, 5.22e-8, 6.46e-8, 7.75e-8, 9.03e-8,
02811         1.06e-7, 1.21e-7, 1.37e-7, 1.57e-7, 1.75e-7, 2.01e-7, 2.3e-7,
02812         2.59e-7, 2.95e-7, 3.26e-7, 3.66e-7, 4.05e-7, 4.47e-7, 4.92e-7,
02813         5.34e-7, 5.84e-7, 6.24e-7, 6.67e-7, 7.14e-7, 7.26e-7, 7.54e-7,
02814         7.84e-7, 8.09e-7, 8.42e-7, 8.62e-7, 8.87e-7, 9.11e-7, 9.36e-7,
02815         9.76e-7, 1.03e-6, 1.11e-6, 1.23e-6, 1.39e-6, 1.61e-6, 1.76e-6,
02816         1.94e-6, 1.97e-6, 1.87e-6, 1.75e-6, 1.56e-6, 1.42e-6, 1.35e-6,
02817         1.32e-6, 1.29e-6, 1.29e-6, 1.29e-6, 1.3e-6, 1.32e-6, 1.33e-6,
02818         1.34e-6, 1.35e-6, 1.33e-6, 1.31e-6, 1.29e-6, 1.24e-6, 1.2e-6,
02819         1.16e-6, 1.1e-6, 1.04e-6, 9.96e-7, 9.38e-7, 8.63e-7, 7.98e-7,
02820         7.26e-7, 6.55e-7, 5.94e-7, 5.35e-7, 4.74e-7, 4.24e-7, 3.77e-7,
02821         3.33e-7, 2.96e-7, 2.63e-7, 2.34e-7, 2.08e-7, 1.85e-7, 1.67e-7,
02822         1.47e-7, 1.32e-7, 1.2e-7, 1.09e-7, 9.85e-8, 9.08e-8, 8.18e-8,
02823         7.56e-8, 6.85e-8, 6.14e-8, 5.83e-8, 5.77e-8, 5e-8, 4.32e-8, 0.
02824     };
02825
02826     static double betaa[98] = { 802., 802., 761., 722., 679., 646., 609., 562.,
02827         511., 472., 436., 406., 377., 355., 338., 319., 299., 278., 255.,
02828         233., 208., 184., 149., 107., 66., 25., -13., -49., -82., -104.,
02829         -119., -130., -139., -144., -146., -146., -147., -148., -150.,
02830         -153., -160., -169., -181., -189., -195., -200., -205., -209.,
02831         -211., -210., -210., -209., -205., -199., -190., -180., -168.,
02832         -157., -143., -126., -108., -89., -63., -32., 1., 35., 65., 95.,
02833         121., 141., 152., 161., 164., 164., 161., 155., 148., 143., 137.,
02834         133., 131., 133., 139., 150., 165., 187., 213., 248., 284., 321.,
02835         372., 449., 514., 569., 609., 642., 673., 673.
02836     };
02837
02838     static double nua[98] = { 2120., 2125., 2130., 2135., 2140., 2145., 2150.,
02839         2155., 2160., 2165., 2170., 2175., 2180., 2185., 2190., 2195.,
02840         2200., 2205., 2210., 2215., 2220., 2225., 2230., 2235., 2240.,
02841         2245., 2250., 2255., 2260., 2265., 2270., 2275., 2280., 2285.,
02842         2290., 2295., 2300., 2305., 2310., 2315., 2320., 2325., 2330.,
02843         2335., 2340., 2345., 2350., 2355., 2360., 2365., 2370., 2375.,
02844         2380., 2385., 2390., 2395., 2400., 2405., 2410., 2415., 2420.,
02845         2425., 2430., 2435., 2440., 2445., 2450., 2455., 2460., 2465.,
02846         2470., 2475., 2480., 2485., 2490., 2495., 2500., 2505., 2510.,
02847         2515., 2520., 2525., 2530., 2535., 2540., 2545., 2550., 2555.,
02848         2560., 2565., 2570., 2575., 2580., 2585., 2590., 2595., 2600., 2605.
02849     };
02850
02851     const double q_n2 = 0.79, t0 = 273.0, tr = 296.0;
02852
02853     /* Check wavenumber range... */
02854     if (nu < nua[0] || nu > nua[97])
02855         return 0;
02856
02857     /* Interpolate B and beta... */
02858     int idx = locate_reg(nua, 98, nu);
02859     double b = LIN(nua[idx], ba[idx], nua[idx + 1], ba[idx + 1], nu);
02860     double beta = LIN(nua[idx], betaa[idx], nua[idx + 1], betaa[idx + 1], nu);
02861
02862     /* Compute absorption coefficient... */
02863     return 0.1 * POW2(p / P0 * t0 / t) * exp(beta * (1 / tr - 1 / t))
02864         * q_n2 * b * (q_n2 + (1 - q_n2) * (1.294 - 0.4545 * t / tr));
02865 }
02866
02867 /*****
02868
02869 double ctmo2(
02870     double nu,
02871     double p,
02872     double t) {
02873
02874     static double ba[90] = { 0., .061, .074, .084, .096, .12, .162, .208, .246,

```

```

02875     .285, .314, .38, .444, .5, .571, .673, .768, .853, .966, 1.097,
02876     1.214, 1.333, 1.466, 1.591, 1.693, 1.796, 1.922, 2.037, 2.154,
02877     2.264, 2.375, 2.508, 2.671, 2.847, 3.066, 3.417, 3.828, 4.204,
02878     4.453, 4.599, 4.528, 4.284, 3.955, 3.678, 3.477, 3.346, 3.29,
02879     3.251, 3.231, 3.226, 3.212, 3.192, 3.108, 3.033, 2.911, 2.798,
02880     2.646, 2.508, 2.322, 2.13, 1.928, 1.757, 1.588, 1.417, 1.253,
02881     1.109, .99, .888, .791, .678, .587, .524, .464, .403, .357, .32,
02882     .29, .267, .242, .215, .182, .16, .146, .128, .103, .087, .081,
02883     .071, .064, 0.
02884 };
02885
02886 static double betaa[90] = { 467., 467., 400., 315., 379., 368., 475., 521.,
02887     531., 512., 442., 444., 430., 381., 335., 324., 296., 248., 215.,
02888     193., 158., 127., 101., 71., 31., -6., -26., -47., -63., -79.,
02889     -88., -88., -87., -90., -98., -99., -109., -134., -160., -167.,
02890     -164., -158., -153., -151., -156., -166., -168., -173., -170.,
02891     -161., -145., -126., -108., -84., -59., -29., 4., 41., 73., 97.,
02892     123., 159., 198., 220., 242., 256., 281., 311., 334., 319., 313.,
02893     321., 323., 310., 315., 320., 335., 361., 378., 373., 338., 319.,
02894     346., 322., 291., 290., 350., 371., 504., 504.
02895 };
02896
02897 static double nua[90] = { 1360., 1365., 1370., 1375., 1380., 1385., 1390.,
02898     1395., 1400., 1405., 1410., 1415., 1420., 1425., 1430., 1435.,
02899     1440., 1445., 1450., 1455., 1460., 1465., 1470., 1475., 1480.,
02900     1485., 1490., 1495., 1500., 1505., 1510., 1515., 1520., 1525.,
02901     1530., 1535., 1540., 1545., 1550., 1555., 1560., 1565., 1570.,
02902     1575., 1580., 1585., 1590., 1595., 1600., 1605., 1610., 1615.,
02903     1620., 1625., 1630., 1635., 1640., 1645., 1650., 1655., 1660.,
02904     1665., 1670., 1675., 1680., 1685., 1690., 1695., 1700., 1705.,
02905     1710., 1715., 1720., 1725., 1730., 1735., 1740., 1745., 1750.,
02906     1755., 1760., 1765., 1770., 1775., 1780., 1785., 1790., 1795.,
02907     1800., 1805.
02908 };
02909
02910 const double q_o2 = 0.21, t0 = 273, tr = 296;
02911
02912 /* Check wavenumber range... */
02913 if (nu < nua[0] || nu > nua[89])
02914     return 0;
02915
02916 /* Interpolate B and beta... */
02917 int idx = locate_reg(nua, 90, nu);
02918 double b = LIN(nua[idx], ba[idx], nua[idx + 1], ba[idx + 1], nu);
02919 double beta = LIN(nua[idx], betaa[idx], nua[idx + 1], betaa[idx + 1], nu);
02920
02921 /* Compute absorption coefficient... */
02922 return 0.1 * POW2(p / P0 * t0 / t) * exp(beta * (1 / tr - 1 / t)) * q_o2 *
02923     b;
02924 }
02925
02926 /*****
02927 void copy_atm(
02928     ctl_t * ctl,
02929     atm_t * atm_dest,
02930     atm_t * atm_src,
02931     int init) {
02932
02933     /* Data size... */
02934     size_t s = (size_t) atm_src->np * sizeof(double);
02935
02936     /* Copy data... */
02937     atm_dest->np = atm_src->np;
02938     memcpy(atm_dest->time, atm_src->time, s);
02939     memcpy(atm_dest->z, atm_src->z, s);
02940     memcpy(atm_dest->lon, atm_src->lon, s);
02941     memcpy(atm_dest->lat, atm_src->lat, s);
02942     memcpy(atm_dest->p, atm_src->p, s);
02943     memcpy(atm_dest->t, atm_src->t, s);
02944     for (int ig = 0; ig < ctl->ng; ig++)
02945         memcpy(atm_dest->q[ig], atm_src->q[ig], s);
02946     for (int iw = 0; iw < ctl->nw; iw++)
02947         memcpy(atm_dest->k[iw], atm_src->k[iw], s);
02948     atm_dest->clz = atm_src->clz;
02949     atm_dest->cldz = atm_src->cldz;
02950     for (int icl = 0; icl < ctl->ncl; icl++)
02951         atm_dest->clk[icl] = atm_src->clk[icl];
02952     atm_dest->sfz = atm_src->sfz;
02953     atm_dest->sfp = atm_src->sfp;
02954     atm_dest->sft = atm_src->sft;
02955     for (int isf = 0; isf < ctl->nsf; isf++)
02956         atm_dest->sfeps[isf] = atm_src->sfeps[isf];
02957
02958     /* Initialize... */
02959     if (init)
02960         for (int ip = 0; ip < atm_dest->np; ip++) {

```

```

02962     atm_dest->p[ip] = 0;
02963     atm_dest->t[ip] = 0;
02964     for (int ig = 0; ig < ctl->ng; ig++)
02965         atm_dest->q[ig][ip] = 0;
02966     for (int iw = 0; iw < ctl->nw; iw++)
02967         atm_dest->k[iw][ip] = 0;
02968     atm_dest->clz = 0;
02969     atm_dest->cldz = 0;
02970     for (int icl = 0; icl < ctl->ncl; icl++)
02971         atm_dest->clk[icl] = 0;
02972     atm_dest->sfz = 0;
02973     atm_dest->sfp = 0;
02974     atm_dest->sft = 0;
02975     for (int isf = 0; isf < ctl->nsf; isf++)
02976         atm_dest->sfeps[isf] = 1;
02977 }
02978 }
02979
02980 /*****
02981
02982 void copy_obs(
02983     ctl_t * ctl,
02984     obs_t * obs_dest,
02985     obs_t * obs_src,
02986     int init) {
02987     /* Data size... */
02988     size_t s = (size_t) obs_src->nr * sizeof(double);
02989
02990     /* Copy data... */
02991     obs_dest->nr = obs_src->nr;
02992     memcpy(obs_dest->time, obs_src->time, s);
02993     memcpy(obs_dest->obsz, obs_src->obsz, s);
02994     memcpy(obs_dest->obslon, obs_src->obslon, s);
02995     memcpy(obs_dest->obslat, obs_src->obslat, s);
02996     memcpy(obs_dest->vpz, obs_src->vpz, s);
02997     memcpy(obs_dest->vplon, obs_src->vplon, s);
02998     memcpy(obs_dest->vplat, obs_src->vplat, s);
02999     memcpy(obs_dest->tpz, obs_src->tpz, s);
03000     memcpy(obs_dest->tplon, obs_src->tplon, s);
03001     memcpy(obs_dest->tplat, obs_src->tplat, s);
03002     for (int id = 0; id < ctl->nd; id++)
03003         memcpy(obs_dest->rad[id], obs_src->rad[id], s);
03004     for (int id = 0; id < ctl->nd; id++)
03005         memcpy(obs_dest->tau[id], obs_src->tau[id], s);
03006
03007     /* Initialize... */
03008     if (init)
03009         for (int id = 0; id < ctl->nd; id++)
03010             for (int ir = 0; ir < obs_dest->nr; ir++)
03011                 if (gsl_finite(obs_dest->rad[id][ir])) {
03012                     obs_dest->rad[id][ir] = 0;
03013                     obs_dest->tau[id][ir] = 0;
03014                 }
03015 }
03016 }
03017
03018 /*****
03019
03020 int find_emitter(
03021     ctl_t * ctl,
03022     const char *emitter) {
03023     for (int ig = 0; ig < ctl->ng; ig++)
03024         if (strcasecmp(ctl->emitter[ig], emitter) == 0)
03025             return ig;
03026     return -1;
03027 }
03028 }
03029 }
03030
03031 /*****
03032
03033 void formod(
03034     ctl_t * ctl,
03035     atm_t * atm,
03036     obs_t * obs) {
03037     int *mask;
03038
03039     /* Allocate... */
03040     ALLOC(mask, int,
03041           ND * NR);
03042
03043     /* Save observation mask... */
03044     for (int id = 0; id < ctl->nd; id++)
03045         for (int ir = 0; ir < obs->nr; ir++)
03046             mask[id * NR + ir] = !gsl_finite(obs->rad[id][ir]);
03047 }
03048

```

```

03049  /* Hydrostatic equilibrium... */
03050  hydrostatic(ctl, atm);
03051
03052  /* EGA forward model... */
03053  if (ctl->formod == 1)
03054      for (int ir = 0; ir < obs->nr; ir++)
03055          formod_pencil(ctl, atm, obs, ir);
03056
03057  /* Call RFM... */
03058  else if (ctl->formod == 2)
03059      formod_rfm(ctl, atm, obs);
03060
03061  /* Apply field-of-view convolution... */
03062  formod_fov(ctl, obs);
03063
03064  /* Convert radiance to brightness temperature... */
03065  if (ctl->write_bbt)
03066      for (int id = 0; id < ctl->nd; id++)
03067          for (int ir = 0; ir < obs->nr; ir++)
03068              obs->rad[id][ir] = brightness(obs->rad[id][ir], ctl->nu[id]);
03069
03070  /* Apply observation mask... */
03071  for (int id = 0; id < ctl->nd; id++)
03072      for (int ir = 0; ir < obs->nr; ir++)
03073          if (mask[id * NR + ir])
03074              obs->rad[id][ir] = GSL_NAN;
03075
03076  /* Free... */
03077  free(mask);
03078 }
03079
03080 /*****
03081
03082 void formod_continua(
03083     ctl_t * ctl,
03084     los_t * los,
03085     int ip,
03086     double *beta) {
03087
03088     static int ig_co2 = -999, ig_h2o = -999;
03089
03090     /* Extinction... */
03091     for (int id = 0; id < ctl->nd; id++)
03092         beta[id] = los->k[ip][id];
03093
03094     /* CO2 continuum... */
03095     if (ctl->ctm_co2) {
03096         if (ig_co2 == -999)
03097             ig_co2 = find_emitter(ctl, "CO2");
03098         if (ig_co2 >= 0)
03099             for (int id = 0; id < ctl->nd; id++)
03100                 beta[id] += ctmco2(ctl->nu[id], los->p[ip], los->t[ip],
03101                                     los->u[ip][ig_co2]) / los->ds[ip];
03102     }
03103
03104     /* H2O continuum... */
03105     if (ctl->ctm_h2o) {
03106         if (ig_h2o == -999)
03107             ig_h2o = find_emitter(ctl, "H2O");
03108         if (ig_h2o >= 0)
03109             for (int id = 0; id < ctl->nd; id++)
03110                 beta[id] += ctmh2o(ctl->nu[id], los->p[ip], los->t[ip],
03111                                     los->q[ip][ig_h2o], los->u[ip][ig_h2o])
03112                                     / los->ds[ip];
03113     }
03114
03115     /* N2 continuum... */
03116     if (ctl->ctm_n2)
03117         for (int id = 0; id < ctl->nd; id++)
03118             beta[id] += ctmn2(ctl->nu[id], los->p[ip], los->t[ip]);
03119
03120     /* O2 continuum... */
03121     if (ctl->ctm_o2)
03122         for (int id = 0; id < ctl->nd; id++)
03123             beta[id] += ctmo2(ctl->nu[id], los->p[ip], los->t[ip]);
03124 }
03125
03126 /*****
03127
03128 void formod_fov(
03129     ctl_t * ctl,
03130     obs_t * obs) {
03131
03132     static double dz[NSHAPE], w[NSHAPE];
03133
03134     static int init = 0, n;
03135

```

```

03136     obs_t *obs2;
03137
03138     double rad[ND][NR], tau[ND][NR], z[NR];
03139
03140     /* Do not take into account FOV... */
03141     if (ctl->fov[0] == '-')
03142         return;
03143
03144     /* Initialize FOV data... */
03145     if (!init) {
03146         init = 1;
03147         read_shape(ctl->fov, dz, w, &n);
03148     }
03149
03150     /* Allocate... */
03151     ALLOC(obs2, obs_t, 1);
03152
03153     /* Copy observation data... */
03154     copy_obs(ctl, obs2, obs, 0);
03155
03156     /* Loop over ray paths... */
03157     for (int ir = 0; ir < obs->nr; ir++) {
03158
03159         /* Get radiance and transmittance profiles... */
03160         int nz = 0;
03161         for (int ir2 = GSL_MAX(ir - NFOV, 0);
03162              ir2 < GSL_MIN(ir + 1 + NFOV, obs->nr); ir2++)
03163             if (obs->time[ir2] == obs->time[ir]) {
03164                 z[nz] = obs2->vpz[ir2];
03165                 for (int id = 0; id < ctl->nd; id++) {
03166                     rad[id][nz] = obs2->rad[id][ir2];
03167                     tau[id][nz] = obs2->tau[id][ir2];
03168                 }
03169                 nz++;
03170             }
03171         if (nz < 2)
03172             ERRMSG("Cannot apply FOV convolution!");
03173
03174         /* Convolute profiles with FOV... */
03175         double wsum = 0;
03176         for (int id = 0; id < ctl->nd; id++) {
03177             obs->rad[id][ir] = 0;
03178             obs->tau[id][ir] = 0;
03179         }
03180         for (int i = 0; i < n; i++) {
03181             double zfov = obs->vpz[ir] + dz[i];
03182             int idx = locate_irr(z, nz, zfov);
03183             for (int id = 0; id < ctl->nd; id++) {
03184                 obs->rad[id][ir] += w[i]
03185                     * LIN(z[idx], rad[id][idx], z[idx + 1], rad[id][idx + 1], zfov);
03186                 obs->tau[id][ir] += w[i]
03187                     * LIN(z[idx], tau[id][idx], z[idx + 1], tau[id][idx + 1], zfov);
03188             }
03189             wsum += w[i];
03190         }
03191         for (int id = 0; id < ctl->nd; id++) {
03192             obs->rad[id][ir] /= wsum;
03193             obs->tau[id][ir] /= wsum;
03194         }
03195     }
03196
03197     /* Free... */
03198     free(obs2);
03199 }
03200
03201 /*****
03202
03203 void formod_pencil(
03204     ctl_t *ctl,
03205     atm_t *atm,
03206     obs_t *obs,
03207     int ir) {
03208
03209     static tbl_t *tbl;
03210
03211     static int init = 0;
03212
03213     los_t *los;
03214
03215     double beta_ctm[ND], rad[ND], tau[ND], tau_refl[ND],
03216         tau_path[ND][NG], tau_gas[ND], x0[3], x1[3];
03217
03218     /* Initialize look-up tables... */
03219     if (!init) {
03220         init = 1;
03221         ALLOC(tbl, tbl_t, 1);
03222         read_tbl(ctl, tbl);

```

```

03223     init_srcfunc(ctl, tbl);
03224 }
03225
03226 /* Allocate... */
03227 ALLOC(los, los_t, 1);
03228
03229 /* Initialize... */
03230 for (int id = 0; id < ctl->nd; id++) {
03231     rad[id] = 0;
03232     tau[id] = 1;
03233     for (int ig = 0; ig < ctl->ng; ig++)
03234         tau_path[id][ig] = 1;
03235 }
03236
03237 /* Raytracing... */
03238 raytrace(ctl, atm, obs, los, ir);
03239
03240 /* Loop over LOS points... */
03241 for (int ip = 0; ip < los->np; ip++) {
03242
03243     /* Get trace gas transmittance... */
03244     intpol_tbl(ctl, tbl, los, ip, tau_path, tau_gas);
03245
03246     /* Get continuum absorption... */
03247     formod_continua(ctl, los, ip, beta_ctm);
03248
03249     /* Compute Planck function... */
03250     formod_srcfunc(ctl, tbl, los->t[ip], los->src[ip]);
03251
03252     /* Loop over channels... */
03253     for (int id = 0; id < ctl->nd; id++)
03254         if (tau_gas[id] > 0) {
03255
03256             /* Get segment emissivity... */
03257             los->eps[ip][id] = 1 - tau_gas[id] * exp(-beta_ctm[id] * los->ds[ip]);
03258
03259             /* Compute radiance... */
03260             rad[id] += los->src[ip][id] * los->eps[ip][id] * tau[id];
03261
03262             /* Compute path transmittance... */
03263             tau[id] *= (1 - los->eps[ip][id]);
03264         }
03265     }
03266
03267 /* Check whether LOS hit the ground... */
03268 if (ctl->sftype >= 1 && los->sft > 0) {
03269
03270     /* Add surface emissions... */
03271     double src_sf[ND];
03272     formod_srcfunc(ctl, tbl, los->sft, src_sf);
03273     for (int id = 0; id < ctl->nd; id++)
03274         rad[id] += los->sfeps[id] * src_sf[id] * tau[id];
03275
03276     /* Check reflectivity... */
03277     int refl = 0;
03278     if (ctl->sftype >= 2)
03279         for (int id = 0; id < ctl->nd; id++)
03280             if (los->sfeps[id] < 1) {
03281                 refl = 1;
03282                 break;
03283             }
03284
03285     /* Calculate reflection... */
03286     if (refl) {
03287
03288         /* Initialize... */
03289         for (int id = 0; id < ctl->nd; id++)
03290             tau_refl[id] = 1;
03291
03292         /* Add down-welling radiance... */
03293         for (int ip = los->np - 1; ip >= 0; ip--)
03294             for (int id = 0; id < ctl->nd; id++) {
03295                 rad[id] += los->src[ip][id] * los->eps[ip][id] * tau_refl[id]
03296                     * tau[id] * (1 - los->sfeps[id]);
03297                 tau_refl[id] *= (1 - los->eps[ip][id]);
03298             }
03299
03300         /* Add solar term... */
03301         if (ctl->sftype >= 3) {
03302
03303             /* Get solar zenith angle... */
03304             double sza2;
03305             if (ctl->sfsza < 0)
03306                 sza2 =
03307                     sza(obs->time[ir], los->lon[los->np - 1], los->lat[los->np - 1]);
03308             else
03309                 sza2 = ctl->sfsza;

```

```

03310
03311     /* Check solar zenith angle... */
03312     if (sza2 < 89.999) {
03313
03314         /* Get angle of incidence... */
03315         geo2cart(los->z[los->np - 1], los->lon[los->np - 1],
03316                 los->lat[los->np - 1], x0);
03317         geo2cart(los->z[0], los->lon[0], los->lat[0], x1);
03318         for (int i = 0; i < 3; i++)
03319             x1[i] -= x0[i];
03320         double cosa = DOTP(x0, x1) / NORM(x0) / NORM(x1);
03321
03322         /* Get ratio of SZA and incident radiation... */
03323         double rcos = cosa / cos(sza2 * M_PI / 180.);
03324
03325         /* Add solar radiation... */
03326         for (int id = 0; id < ctl->nd; id++)
03327             rad[id] += 6.764e-5 / (2. * M_PI) * planck(TSUN, ctl->nu[id])
03328                 * tau_refl[id] * (1 - los->sfeps[id]) * tau[id] * rcos;
03329     }
03330 }
03331 }
03332 }
03333
03334 /* Copy results... */
03335 for (int id = 0; id < ctl->nd; id++) {
03336     obs->rad[id][ir] = rad[id];
03337     obs->tau[id][ir] = tau[id];
03338 }
03339
03340 /* Free... */
03341 free(los);
03342 }
03343
03344 /*****
03345
03346 void formod_rfm(
03347     ctl_t * ctl,
03348     atm_t * atm,
03349     obs_t * obs) {
03350
03351     los_t *los;
03352
03353     FILE *out;
03354
03355     char cmd[2 * LEN], filename[2 * LEN],
03356           rfmflg[LEN] = { "RAD TRA MIX LIN SFC" };
03357
03358     double f[NSHAPE], nu[NSHAPE], nu0, nu1, obsz = -999, tsurf,
03359           xd[3], xo[3], xv[3], z[NR], zmin, zmax;
03360
03361     int i, id, ig, ip, ir, iw, n, nadir = 0;
03362
03363     /* Allocate... */
03364     ALLOC(los, los_t, 1);
03365
03366     /* Check observer positions... */
03367     for (ir = 1; ir < obs->nr; ir++)
03368         if (obs->obsz[ir] != obs->obsz[0]
03369             || obs->obslon[ir] != obs->obslon[0]
03370             || obs->obslat[ir] != obs->obslat[0])
03371             ERRMSG("RFM interface requires identical observer positions!");
03372
03373     /* Check extinction data... */
03374     for (iw = 0; iw < ctl->nw; iw++)
03375         for (ip = 0; ip < atm->np; ip++)
03376             if (atm->k[iw][ip] != 0)
03377                 ERRMSG("RFM interface cannot handle extinction data!");
03378
03379     /* Get altitude range of atmospheric data... */
03380     gsl_stats_minmax(&zmin, &zmax, atm->z, 1, (size_t) atm->np);
03381
03382     /* Observer within atmosphere? */
03383     if (obs->obsz[0] >= zmin && obs->obsz[0] <= zmax) {
03384         obsz = obs->obsz[0];
03385         strcat(rfmflg, " OBS");
03386     }
03387
03388     /* Determine tangent altitude or air mass factor... */
03389     for (ir = 0; ir < obs->nr; ir++) {
03390
03391         /* Raytracing... */
03392         raytrace(ctl, atm, obs, los, ir);
03393
03394         /* Nadir? */
03395         if (obs->tpz[ir] <= zmin) {
03396             geo2cart(obs->obsz[ir], obs->obslon[ir], obs->obslat[ir], xo);

```

```

03397     geo2cart(obs->vpz[ir], obs->vplon[ir], obs->vplat[ir], xv);
03398     for (i = 0; i < 3; i++)
03399         xd[i] = xo[i] - xv[i];
03400     z[ir] = NORM(xo) * NORM(xd) / DOTP(xo, xd);
03401     nadir++;
03402     } else
03403         z[ir] = obs->tpz[ir];
03404     }
03405     if (nadir > 0 && nadir < obs->nr)
03406         ERRMSG("Limb and nadir not simultaneously possible!");
03407
03408     /* Nadir? */
03409     if (nadir)
03410         strcat(rfmflg, " NAD");
03411
03412     /* Get surface temperature... */
03413     tsurf = atm->t[gsl_stats_min_index(atm->z, 1, (size_t) atm->np)];
03414
03415     /* Refraction? */
03416     if (!nadir && !ctl->refrac)
03417         strcat(rfmflg, " GEO");
03418
03419     /* Continua? */
03420     if (ctl->ctm_co2 || ctl->ctm_h2o || ctl->ctm_n2 || ctl->ctm_o2)
03421         strcat(rfmflg, " CTM");
03422
03423     /* Write atmospheric data file... */
03424     write_atm_rfm("rfm.atm", ctl, atm);
03425
03426     /* Loop over channels... */
03427     for (id = 0; id < ctl->nd; id++) {
03428
03429         /* Read filter function... */
03430         sprintf(filename, "%s_%.4f.filt", ctl->tblbase, ctl->nu[id]);
03431         read_shape(filename, nu, f, &n);
03432
03433         /* Set spectral range... */
03434         nu0 = nu[0];
03435         nu1 = nu[n - 1];
03436
03437         /* Create RFM driver file... */
03438         if (!(out = fopen("rfm.drv", "w")))
03439             ERRMSG("Cannot create file!");
03440         fprintf(out, "*HDR\nRFM call by JURASSIC.\n");
03441         fprintf(out, "*FLG\n%s\n", rfmflg);
03442         fprintf(out, "*SPC\n%.4f %.4f 0.0005\n", nu0, nu1);
03443         fprintf(out, "*GAS\n");
03444         for (ig = 0; ig < ctl->ng; ig++)
03445             fprintf(out, "%s\n", ctl->emitter[ig]);
03446         fprintf(out, "*ATM\nrfm.atm\n");
03447         fprintf(out, "*TAN\n");
03448         for (ir = 0; ir < obs->nr; ir++)
03449             fprintf(out, "%g\n", z[ir]);
03450         fprintf(out, "*SFC\n%g 1.0\n", tsurf);
03451         if (obsz >= 0)
03452             fprintf(out, "*OBS\n%g\n", obsz);
03453         fprintf(out, "*HIT\n%s\n", ctl->rfmhit);
03454         fprintf(out, "*XSC\n");
03455         for (ig = 0; ig < ctl->ng; ig++)
03456             if (ctl->rfmxsc[ig][0] != '-')
03457                 fprintf(out, "%s\n", ctl->rfmxsc[ig]);
03458         fprintf(out, "*END\n");
03459         fclose(out);
03460
03461         /* Remove temporary files... */
03462         if (system("rm -f rfm.runlog rad_*.asc tra_*.asc"))
03463             ERRMSG("Cannot remove temporary files!");
03464
03465         /* Call RFM... */
03466         sprintf(cmd, "echo | %s", ctl->rfmbin);
03467         if (system(cmd))
03468             ERRMSG("Error while calling RFM!");
03469
03470         /* Read data... */
03471         for (ir = 0; ir < obs->nr; ir++) {
03472             obs->rad[id][ir] = read_obs_rfm("rad", z[ir], nu, f, n) * 1e-5;
03473             obs->tau[id][ir] = read_obs_rfm("tra", z[ir], nu, f, n);
03474         }
03475     }
03476
03477     /* Remove temporary files... */
03478     if (system("rm -f rfm.drv rfm.atm rfm.runlog rad_*.asc tra_*.asc"))
03479         ERRMSG("Error while removing temporary files!");
03480
03481     /* Free... */
03482     free(los);
03483 }

```



```

03484
03485 /*****
03486
03487 void formod_srcfunc(
03488     ctl_t * ctl,
03489     tbl_t * tbl,
03490     double t,
03491     double *src) {
03492
03493     /* Determine index in temperature array... */
03494     int it = locate_reg(tbl->st, TBLNS, t);
03495
03496     /* Interpolate Planck function value... */
03497     for (int id = 0; id < ctl->nd; id++)
03498         src[id] = LIN(tbl->st[it], tbl->sr[it][id],
03499                     tbl->st[it + 1], tbl->sr[it + 1][id], t);
03500 }
03501
03502 /*****
03503
03504 void geo2cart(
03505     double z,
03506     double lon,
03507     double lat,
03508     double *x) {
03509
03510     double radius = z + RE;
03511
03512     x[0] = radius * cos(lat / 180 * M_PI) * cos(lon / 180 * M_PI);
03513     x[1] = radius * cos(lat / 180 * M_PI) * sin(lon / 180 * M_PI);
03514     x[2] = radius * sin(lat / 180 * M_PI);
03515 }
03516
03517 /*****
03518
03519 void hydrostatic(
03520     ctl_t * ctl,
03521     atm_t * atm) {
03522
03523     const double mmair = 28.96456e-3, mmh2o = 18.0153e-3;
03524
03525     const int ipts = 20;
03526
03527     static int ig_h2o = -999;
03528
03529     double dzmin = 1e99, e = 0;
03530
03531     int ipref = 0;
03532
03533     /* Check reference height... */
03534     if (ctl->hydz < 0)
03535         return;
03536
03537     /* Determine emitter index of H2O... */
03538     if (ig_h2o == -999)
03539         ig_h2o = find_emitter(ctl, "H2O");
03540
03541     /* Find air parcel next to reference height... */
03542     for (int ip = 0; ip < atm->np; ip++)
03543         if (fabs(atm->z[ip] - ctl->hydz) < dzmin) {
03544             dzmin = fabs(atm->z[ip] - ctl->hydz);
03545             ipref = ip;
03546         }
03547
03548     /* Upper part of profile... */
03549     for (int ip = ipref + 1; ip < atm->np; ip++) {
03550         double mean = 0;
03551         for (int i = 0; i < ipts; i++) {
03552             if (ig_h2o >= 0)
03553                 e = LIN(0.0, atm->q[ig_h2o][ip - 1],
03554                       ipts - 1.0, atm->q[ig_h2o][ip], (double) i);
03555             mean += (e * mmh2o + (1 - e) * mmair)
03556                   * G0 / RI
03557                   / LIN(0.0, atm->t[ip - 1], ipts - 1.0, atm->t[ip], (double) i) / ipts;
03558         }
03559
03560         /* Compute p(z,T)... */
03561         atm->p[ip] =
03562             exp(log(atm->p[ip - 1]) - mean * 1000 * (atm->z[ip] - atm->z[ip - 1]));
03563     }
03564
03565     /* Lower part of profile... */
03566     for (int ip = ipref - 1; ip >= 0; ip--) {
03567         double mean = 0;
03568         for (int i = 0; i < ipts; i++) {
03569             if (ig_h2o >= 0)
03570                 e = LIN(0.0, atm->q[ig_h2o][ip + 1],

```

```

03571         ipt_s - 1.0, atm->q[ig_h2o][ip], (double) i);
03572     mean += (e * mmh2o + (1 - e) * mmair)
03573         * G0 / RI
03574         / LIN(0.0, atm->t[ip + 1], ipt_s - 1.0, atm->t[ip], (double) i) / ipt_s;
03575 }
03576
03577 /* Compute p(z,T)... */
03578 atm->p[ip] =
03579     exp(log(atm->p[ip + 1]) - mean * 1000 * (atm->z[ip] - atm->z[ip + 1]));
03580 }
03581 }
03582
03583 /*****
03584
03585 void idx2name(
03586     ctl_t * ctl,
03587     int idx,
03588     char *quantity) {
03589
03590     if (idx == IDXP)
03591         sprintf(quantity, "PRESSURE");
03592
03593     if (idx == IDXT)
03594         sprintf(quantity, "TEMPERATURE");
03595
03596     for (int ig = 0; ig < ctl->ng; ig++)
03597         if (idx == IDXQ(ig))
03598             sprintf(quantity, "%s", ctl->emitter[ig]);
03599
03600     for (int iw = 0; iw < ctl->nw; iw++)
03601         if (idx == IDXK(iw))
03602             sprintf(quantity, "EXTINCT_WINDOW_%d", iw);
03603
03604     if (idx == IDXCLZ)
03605         sprintf(quantity, "CLOUD_HEIGHT");
03606
03607     if (idx == IDXCLDZ)
03608         sprintf(quantity, "CLOUD_DEPTH");
03609
03610     for (int icl = 0; icl < ctl->ncl; icl++)
03611         if (idx == IDXCLK(icl))
03612             sprintf(quantity, "CLOUD_EXTINCT_%.4f", ctl->clnu[icl]);
03613
03614     if (idx == IDXSFZ)
03615         sprintf(quantity, "SURFACE_HEIGHT");
03616
03617     if (idx == IDXSFPS)
03618         sprintf(quantity, "SURFACE_PRESSURE");
03619
03620     if (idx == IDXSFST)
03621         sprintf(quantity, "SURFACE_TEMPERATURE");
03622
03623     for (int isf = 0; isf < ctl->nsf; isf++)
03624         if (idx == IDXSFEPS(isf))
03625             sprintf(quantity, "SURFACE_EMISSIVITY_%.4f", ctl->sfnu[isf]);
03626 }
03627
03628 /*****
03629
03630 void init_srcfunc(
03631     ctl_t * ctl,
03632     tbl_t * tbl) {
03633
03634     char filename[2 * LEN];
03635
03636     double f[NSHAPE], nu[NSHAPE];
03637
03638     int n;
03639
03640     /* Write info... */
03641     LOG(1, "Initialize source function table...");
03642
03643     /* Loop over channels... */
03644     for (int id = 0; id < ctl->nd; id++) {
03645
03646         /* Read filter function... */
03647         sprintf(filename, "%s_%.4f.filt", ctl->tblbase, ctl->nu[id]);
03648         read_shape(filename, nu, f, &n);
03649
03650         /* Get minimum grid spacing... */
03651         double dnu = 1.0;
03652         for (int i = 1; i < n; i++)
03653             dnu = GSL_MIN(dnu, nu[i] - nu[i - 1]);
03654
03655         /* Compute source function table... */
03656         #pragma omp parallel for default(none) shared(ctl,tbl,id,nu,f,n,dnu)
03657         for (int it = 0; it < TBLNS; it++) {

```

```

03658
03659 /* Set temperature... */
03660 tbl->st[it] = LIN(0.0, TMIN, TBLNS - 1.0, TMAX, (double) it);
03661
03662 /* Integrate Planck function... */
03663 double fsum = tbl->sr[it][id] = 0;
03664 for (double fnu = nu[0]; fnu <= nu[n - 1]; fnu += dnu) {
03665     int i = locate_irr(nu, n, fnu);
03666     double ff = LIN(nu[i], f[i], nu[i + 1], f[i + 1], fnu);
03667     fsum += ff;
03668     tbl->sr[it][id] += ff * planck(tbl->st[it], fnu);
03669 }
03670 tbl->sr[it][id] /= fsum;
03671 }
03672 }
03673 }
03674
03675 /*****
03676 void intpol_atm(
03677     ctl_t * ctl,
03678     atm_t * atm,
03679     double z,
03680     double *p,
03681     double *t,
03682     double *q,
03683     double *k) {
03684
03685     /* Get array index... */
03686     int ip = locate_irr(atm->z, atm->np, z);
03687
03688     /* Interpolate... */
03689     *p = EXP(atm->z[ip], atm->p[ip], atm->z[ip + 1], atm->p[ip + 1], z);
03690     *t = LIN(atm->z[ip], atm->t[ip], atm->z[ip + 1], atm->t[ip + 1], z);
03691     for (int ig = 0; ig < ctl->ng; ig++)
03692         q[ig] =
03693             LIN(atm->z[ip], atm->q[ig][ip], atm->z[ip + 1], atm->q[ig][ip + 1], z);
03694     for (int iw = 0; iw < ctl->nw; iw++)
03695         k[iw] =
03696             LIN(atm->z[ip], atm->k[iw][ip], atm->z[ip + 1], atm->k[iw][ip + 1], z);
03697 }
03698
03699 /*****
03700 void intpol_tbl(
03701     ctl_t * ctl,
03702     tbl_t * tbl,
03703     los_t * los,
03704     int ip,
03705     double tau_path[ND][NG],
03706     double tau_seg[ND]) {
03707
03708     double eps, u;
03709
03710     /* Loop over channels... */
03711     for (int id = 0; id < ctl->nd; id++) {
03712
03713         /* Initialize... */
03714         tau_seg[id] = 1;
03715
03716         /* Loop over emitters... */
03717         for (int ig = 0; ig < ctl->ng; ig++) {
03718
03719             /* Check size of table (pressure)... */
03720             if (tbl->np[id][ig] < 30)
03721                 eps = 0;
03722
03723             /* Check transmittance... */
03724             else if (tau_path[id][ig] < 1e-9)
03725                 eps = 1;
03726
03727             /* Interpolate... */
03728             else {
03729
03730                 /* Determine pressure and temperature indices... */
03731                 int ipr = locate_irr(tbl->p[id][ig], tbl->np[id][ig], los->p[ip]);
03732                 int it0 =
03733                     locate_reg(tbl->t[id][ig][ipr], tbl->nt[id][ig][ipr], los->t[ip]);
03734                 int it1 =
03735                     locate_reg(tbl->t[id][ig][ipr + 1], tbl->nt[id][ig][ipr + 1],
03736                             los->t[ip]);
03737
03738                 /* Check size of table (temperature and column density)... */
03739                 if (tbl->nt[id][ig][ipr] < 2 || tbl->nt[id][ig][ipr + 1] < 2
03740                     || tbl->nu[id][ig][ipr][it0] < 2
03741                     || tbl->nu[id][ig][ipr][it0 + 1] < 2
03742                     || tbl->nu[id][ig][ipr + 1][it1] < 2

```

```

03745         || tbl->nu[id][ig][ipr + 1][itl + 1] < 2)
03746         eps = 0;
03747
03748     else {
03749
03750         /* Get emissivities of extended path... */
03751         u = interpol_tbl_u(tbl, ig, id, ipr, it0, 1 - tau_path[id][ig]);
03752         double eps00
03753             = interpol_tbl_eps(tbl, ig, id, ipr, it0, u + los->u[ip][ig]);
03754
03755         u = interpol_tbl_u(tbl, ig, id, ipr, it0 + 1, 1 - tau_path[id][ig]);
03756         double eps01 =
03757             interpol_tbl_eps(tbl, ig, id, ipr, it0 + 1, u + los->u[ip][ig]);
03758
03759         u = interpol_tbl_u(tbl, ig, id, ipr + 1, it1, 1 - tau_path[id][ig]);
03760         double eps10 =
03761             interpol_tbl_eps(tbl, ig, id, ipr + 1, it1, u + los->u[ip][ig]);
03762
03763         u =
03764             interpol_tbl_u(tbl, ig, id, ipr + 1, it1 + 1, 1 - tau_path[id][ig]);
03765         double eps11 =
03766             interpol_tbl_eps(tbl, ig, id, ipr + 1, it1 + 1, u + los->u[ip][ig]);
03767
03768         /* Interpolate with respect to temperature... */
03769         eps00 = LIN(tbl->t[id][ig][ipr][it0], eps00,
03770             tbl->t[id][ig][ipr][it0 + 1], eps01, los->t[ip]);
03771         eps11 = LIN(tbl->t[id][ig][ipr + 1][it1], eps10,
03772             tbl->t[id][ig][ipr + 1][it1 + 1], eps11, los->t[ip]);
03773
03774         /* Interpolate with respect to pressure... */
03775         eps00 = LIN(tbl->p[id][ig][ipr], eps00,
03776             tbl->p[id][ig][ipr + 1], eps11, los->p[ip]);
03777
03778         /* Check emssivity range... */
03779         eps00 = GSL_MAX(GSL_MIN(eps00, 1), 0);
03780
03781         /* Determine segment emissivity... */
03782         eps = 1 - (1 - eps00) / tau_path[id][ig];
03783     }
03784 }
03785
03786 /* Get transmittance of extended path... */
03787 tau_path[id][ig] *= (1 - eps);
03788
03789 /* Get segment transmittance... */
03790 tau_seg[id] *= (1 - eps);
03791 }
03792 }
03793 }
03794
03795 /*****
03796
03797 double interpol_tbl_eps(
03798     tbl_t * tbl,
03799     int ig,
03800     int id,
03801     int ip,
03802     int it,
03803     double u) {
03804
03805     /* Lower boundary... */
03806     if (u < tbl->u[id][ig][ip][it][0])
03807         return LIN(0, 0, tbl->u[id][ig][ip][it][0], tbl->eps[id][ig][ip][it][0],
03808             u);
03809
03810     /* Upper boundary... */
03811     else if (u > tbl->u[id][ig][ip][it][tbl->nu[id][ig][ip][it] - 1])
03812         return LIN(tbl->u[id][ig][ip][it][tbl->nu[id][ig][ip][it] - 1],
03813             tbl->eps[id][ig][ip][it][tbl->nu[id][ig][ip][it] - 1],
03814             1e30, 1, u);
03815
03816     /* Interpolation... */
03817     else {
03818
03819         /* Get index... */
03820         int idx = locate_tbl(tbl->u[id][ig][ip][it], tbl->nu[id][ig][ip][it], u);
03821
03822         /* Interpolate... */
03823         return
03824             LIN(tbl->u[id][ig][ip][it][idx], tbl->eps[id][ig][ip][it][idx],
03825             tbl->u[id][ig][ip][it][idx + 1], tbl->eps[id][ig][ip][it][idx + 1],
03826             u);
03827     }
03828 }
03829
03830 /*****
03831

```

```

03832 double intpol_tbl_u(
03833     tbl_t * tbl,
03834     int ig,
03835     int id,
03836     int ip,
03837     int it,
03838     double eps) {
03839
03840     /* Lower boundary... */
03841     if (eps < tbl->eps[id][ig][ip][it][0])
03842         return LIN(0, 0, tbl->eps[id][ig][ip][it][0], tbl->u[id][ig][ip][it][0],
03843             eps);
03844
03845     /* Upper boundary... */
03846     else if (eps > tbl->eps[id][ig][ip][it][tbl->nu[id][ig][ip][it] - 1])
03847         return LIN(tbl->eps[id][ig][ip][it][tbl->nu[id][ig][ip][it] - 1],
03848             tbl->u[id][ig][ip][it][tbl->nu[id][ig][ip][it] - 1],
03849             1, 1e30, eps);
03850
03851     /* Interpolation... */
03852     else {
03853
03854         /* Get index... */
03855         int idx
03856             = locate_tbl(tbl->eps[id][ig][ip][it], tbl->nu[id][ig][ip][it], eps);
03857
03858         /* Interpolate... */
03859         return
03860             LIN(tbl->eps[id][ig][ip][it][idx], tbl->u[id][ig][ip][it][idx],
03861                 tbl->eps[id][ig][ip][it][idx + 1], tbl->u[id][ig][ip][it][idx + 1],
03862                 eps);
03863     }
03864 }
03865
03866 /*****
03867
03868 void jsec2time(
03869     double jsec,
03870     int *year,
03871     int *mon,
03872     int *day,
03873     int *hour,
03874     int *min,
03875     int *sec,
03876     double *remain) {
03877
03878     struct tm t0, *t1;
03879
03880     t0.tm_year = 100;
03881     t0.tm_mon = 0;
03882     t0.tm_mday = 1;
03883     t0.tm_hour = 0;
03884     t0.tm_min = 0;
03885     t0.tm_sec = 0;
03886
03887     time_t jsec0 = (time_t) jsec + timegm(&t0);
03888     t1 = gmtime(&jsec0);
03889
03890     *year = t1->tm_year + 1900;
03891     *mon = t1->tm_mon + 1;
03892     *day = t1->tm_mday;
03893     *hour = t1->tm_hour;
03894     *min = t1->tm_min;
03895     *sec = t1->tm_sec;
03896     *remain = jsec - floor(jsec);
03897 }
03898
03899 /*****
03900
03901 void kernel(
03902     ctl_t * ctl,
03903     atm_t * atm,
03904     obs_t * obs,
03905     gsl_matrix * k) {
03906
03907     atm_t *atm1;
03908     obs_t *obs1;
03909
03910     gsl_vector *x0, *x1, *yy0, *yy1;
03911
03912     int *iqa;
03913
03914     /* Get sizes... */
03915     size_t m = k->size1;
03916     size_t n = k->size2;
03917
03918     /* Allocate... */

```

```

03919 x0 = gsl_vector_alloc(n);
03920 yy0 = gsl_vector_alloc(m);
03921 ALLOC(iqa, int,
03922       N);
03923
03924 /* Compute radiance for undisturbed atmospheric data... */
03925 formod(ctl, atm, obs);
03926
03927 /* Compose vectors... */
03928 atm2x(ctl, atm, x0, iqa, NULL);
03929 obs2y(ctl, obs, yy0, NULL, NULL);
03930
03931 /* Initialize kernel matrix... */
03932 gsl_matrix_set_zero(k);
03933
03934 /* Loop over state vector elements... */
03935 #pragma omp parallel for default(none) shared(ctl,atm,obs,k,x0,yy0,n,m,iqa) private(x1, yy1, atml,
03936      obs1)
03937 for (size_t j = 0; j < n; j++) {
03938     /* Allocate... */
03939     x1 = gsl_vector_alloc(n);
03940     yy1 = gsl_vector_alloc(m);
03941     ALLOC(atml, atm_t, 1);
03942     ALLOC(obs1, obs_t, 1);
03943
03944     /* Set perturbation size... */
03945     double h;
03946     if (iqa[j] == IDXP)
03947         h = GSL_MAX(fabs(0.01 * gsl_vector_get(x0, j)), 1e-7);
03948     else if (iqa[j] == IDXT)
03949         h = 1.0;
03950     else if (iqa[j] >= IDXQ(0) && iqa[j] < IDXQ(ctl->ng))
03951         h = GSL_MAX(fabs(0.01 * gsl_vector_get(x0, j)), 1e-15);
03952     else if (iqa[j] >= IDXK(0) && iqa[j] < IDXK(ctl->nw))
03953         h = 1e-4;
03954     else if (iqa[j] == IDXCLZ || iqa[j] == IDXCLDZ)
03955         h = 1.0;
03956     else if (iqa[j] >= IDXCLK(0) && iqa[j] < IDXCLK(ctl->ncl))
03957         h = 1e-4;
03958     else if (iqa[j] == IDXSFZ)
03959         h = 0.1;
03960     else if (iqa[j] == IDXSFP)
03961         h = 10.0;
03962     else if (iqa[j] == IDXSFT)
03963         h = 1.0;
03964     else if (iqa[j] >= IDXSFEPS(0) && iqa[j] < IDXSFEPS(ctl->nsf))
03965         h = 1e-2;
03966     else
03967         ERRMSG("Cannot set perturbation size!");
03968
03969     /* Disturb state vector element... */
03970     gsl_vector_memcpy(x1, x0);
03971     gsl_vector_set(x1, j, gsl_vector_get(x1, j) + h);
03972     copy_atm(ctl, atml, atm, 0);
03973     copy_obs(ctl, obs1, obs, 0);
03974     x2atm(ctl, x1, atml);
03975
03976     /* Compute radiance for disturbed atmospheric data... */
03977     formod(ctl, atml, obs1);
03978
03979     /* Compose measurement vector for disturbed radiance data... */
03980     obs2y(ctl, obs1, yy1, NULL, NULL);
03981
03982     /* Compute derivatives... */
03983     for (size_t i = 0; i < m; i++)
03984         gsl_matrix_set(k, i, j,
03985                        (gsl_vector_get(yy1, i) - gsl_vector_get(yy0, i)) / h);
03986
03987     /* Free... */
03988     gsl_vector_free(x1);
03989     gsl_vector_free(yy1);
03990     free(atml);
03991     free(obs1);
03992 }
03993
03994 /* Free... */
03995 gsl_vector_free(x0);
03996 gsl_vector_free(yy0);
03997 free(iqa);
03998 }
03999
04000 /*****
04001
04002 int locate_irr(
04003     double *xx,
04004     int n,

```

```

04005     double x) {
04006
04007     int ilo = 0;
04008     int ihi = n - 1;
04009     int i = (ihi + ilo) » 1;
04010
04011     if (xx[i] < xx[i + 1])
04012         while (ihi > ilo + 1) {
04013             i = (ihi + ilo) » 1;
04014             if (xx[i] > x)
04015                 ihi = i;
04016             else
04017                 ilo = i;
04018         } else
04019         while (ihi > ilo + 1) {
04020             i = (ihi + ilo) » 1;
04021             if (xx[i] <= x)
04022                 ihi = i;
04023             else
04024                 ilo = i;
04025         }
04026
04027     return ilo;
04028 }
04029
04030 /*****
04031
04032 int locate_reg(
04033     double *xx,
04034     int n,
04035     double x) {
04036
04037     /* Calculate index... */
04038     int i = (int) ((x - xx[0]) / (xx[1] - xx[0]));
04039
04040     /* Check range... */
04041     if (i < 0)
04042         return 0;
04043     else if (i > n - 2)
04044         return n - 2;
04045     else
04046         return i;
04047 }
04048
04049 /*****
04050
04051 int locate_tbl(
04052     float *xx,
04053     int n,
04054     double x) {
04055
04056     int ilo = 0;
04057     int ihi = n - 1;
04058     int i = (ihi + ilo) » 1;
04059
04060     while (ihi > ilo + 1) {
04061         i = (ihi + ilo) » 1;
04062         if (xx[i] > x)
04063             ihi = i;
04064         else
04065             ilo = i;
04066     }
04067
04068     return ilo;
04069 }
04070
04071 /*****
04072
04073 size_t obs2y(
04074     ctl_t * ctl,
04075     obs_t * obs,
04076     gsl_vector * y,
04077     int *ida,
04078     int *ira) {
04079
04080     size_t m = 0;
04081
04082     /* Determine measurement vector... */
04083     for (int ir = 0; ir < obs->nr; ir++)
04084         for (int id = 0; id < ctl->nd; id++)
04085             if (gsl_finite(obs->rad[id][ir])) {
04086                 if (y != NULL)
04087                     gsl_vector_set(y, m, obs->rad[id][ir]);
04088                 if (ida != NULL)
04089                     ida[m] = id;
04090                 if (ira != NULL)
04091                     ira[m] = ir;

```

```

04092         m++;
04093     }
04094
04095     return m;
04096 }
04097
04098 /*****
04099
04100 double planck(
04101     double t,
04102     double nu) {
04103
04104     return C1 * POW3(nu) / gsl_expml(C2 * nu / t);
04105 }
04106
04107 /*****
04108
04109 void raytrace(
04110     ctl_t * ctl,
04111     atm_t * atm,
04112     obs_t * obs,
04113     los_t * los,
04114     int ir) {
04115
04116     const double h = 0.02, zrefrac = 60;
04117
04118     double ds, ex0[3], ex1[3], k[NW], lat, lon, n, ng[3], norm,
04119         p, q[NG], t, x[3], xh[3], xobs[3], xvp[3], z = 1e99, zmax, zmin;
04120
04121     int stop = 0;
04122
04123     /* Initialize... */
04124     los->np = 0;
04125     los->sft = -999;
04126     obs->tpz[ir] = obs->vpz[ir];
04127     obs->tplon[ir] = obs->vplon[ir];
04128     obs->tplat[ir] = obs->vplat[ir];
04129
04130     /* Get altitude range of atmospheric data... */
04131     gsl_stats_minmax(&zmin, &zmax, atm->z, 1, (size_t) atm->np);
04132     if (ctl->nsf > 0) {
04133         zmin = GSL_MAX(atm->sfz, zmin);
04134         if (atm->sfp > 0) {
04135             int ip = locate_irr(atm->p, atm->np, atm->sfp);
04136             double zip = LIN(log(atm->p[ip]), atm->z[ip],
04137                 log(atm->p[ip + 1]), atm->z[ip + 1], log(atm->sfp));
04138             zmin = GSL_MAX(zip, zmin);
04139         }
04140     }
04141
04142     /* Check observer altitude... */
04143     if (obs->obsz[ir] < zmin)
04144         ERRMSG("Observer below surface!");
04145
04146     /* Check view point altitude... */
04147     if (obs->vpz[ir] > zmax)
04148         return;
04149
04150     /* Determine Cartesian coordinates for observer and view point... */
04151     geo2cart(obs->obsz[ir], obs->obslon[ir], obs->obslat[ir], xobs);
04152     geo2cart(obs->vpz[ir], obs->vplon[ir], obs->vplat[ir], xvp);
04153
04154     /* Determine initial tangent vector... */
04155     for (int i = 0; i < 3; i++)
04156         ex0[i] = xvp[i] - xobs[i];
04157     norm = NORM(ex0);
04158     for (int i = 0; i < 3; i++)
04159         ex0[i] /= norm;
04160
04161     /* Observer within atmosphere... */
04162     for (int i = 0; i < 3; i++)
04163         x[i] = xobs[i];
04164
04165     /* Observer above atmosphere (search entry point)... */
04166     if (obs->obsz[ir] > zmax) {
04167         double dmax = norm, dmin = 0;
04168         while (fabs(dmin - dmax) > 0.001) {
04169             double d = (dmax + dmin) / 2;
04170             for (int i = 0; i < 3; i++)
04171                 x[i] = xobs[i] + d * ex0[i];
04172             cart2geo(x, &z, &lon, &lat);
04173             if (z <= zmax && z > zmax - 0.001)
04174                 break;
04175             if (z < zmax - 0.0005)
04176                 dmax = d;
04177             else
04178                 dmin = d;

```



```

04179     }
04180 }
04181
04182 /* Ray-tracing... */
04183 while (1) {
04184
04185     /* Set step length... */
04186     ds = ctl->rayds;
04187     if (ctl->raydz > 0) {
04188         norm = NORM(x);
04189         for (int i = 0; i < 3; i++)
04190             xh[i] = x[i] / norm;
04191         double cosa = fabs(DOTP(ex0, xh));
04192         if (cosa != 0)
04193             ds = GSL_MIN(ctl->rayds, ctl->raydz / cosa);
04194     }
04195
04196     /* Determine geolocation... */
04197     cart2geo(x, &z, &lon, &lat);
04198
04199     /* Check if LOS hits the ground or has left atmosphere... */
04200     if (z < zmin || z > zmax) {
04201         stop = (z < zmin ? 2 : 1);
04202         double frac =
04203             ((z <
04204              zmin ? zmin : zmax) - los->z[los->np - 1]) / (z - los->z[los->np -
04205                                                         1]);
04206         geo2cart(los->z[los->np - 1], los->lon[los->np - 1],
04207                 los->lat[los->np - 1], xh);
04208         for (int i = 0; i < 3; i++)
04209             x[i] = xh[i] + frac * (x[i] - xh[i]);
04210         cart2geo(x, &z, &lon, &lat);
04211         los->ds[los->np - 1] = ds * frac;
04212         ds = 0;
04213     }
04214
04215     /* Interpolate atmospheric data... */
04216     intpol_atm(ctl, atm, z, &p, &t, q, k);
04217
04218     /* Save data... */
04219     los->lon[los->np] = lon;
04220     los->lat[los->np] = lat;
04221     los->z[los->np] = z;
04222     los->p[los->np] = p;
04223     los->t[los->np] = t;
04224     for (int ig = 0; ig < ctl->ng; ig++)
04225         los->q[los->np][ig] = q[ig];
04226     for (int id = 0; id < ctl->nd; id++)
04227         los->k[los->np][id] = k[ctl->>window[id]];
04228     los->ds[los->np] = ds;
04229
04230     /* Add cloud extinction... */
04231     if (ctl->ncl > 0 && atm->cldz > 0) {
04232         double aux = exp(-0.5 * POW2((z - atm->clz) / atm->cldz));
04233         for (int id = 0; id < ctl->nd; id++) {
04234             int icl = locate_irr(ctl->clnu, ctl->ncl, ctl->nu[id]);
04235             los->k[los->np][id]
04236                 += aux * LIN(ctl->clnu[icl], atm->clk[icl],
04237                             ctl->clnu[icl + 1], atm->clk[icl + 1], ctl->nu[id]);
04238         }
04239     }
04240
04241     /* Increment and check number of LOS points... */
04242     if ((++los->np) > NLOS)
04243         ERRMSG("Too many LOS points!");
04244
04245     /* Check stop flag... */
04246     if (stop) {
04247
04248         /* Set surface temperature... */
04249         if (ctl->nsf > 0 && atm->sft > 0)
04250             t = atm->sft;
04251         los->sft = (stop == 2 ? t : -999);
04252
04253         /* Set surface emissivity... */
04254         for (int isf = 0; isf < ctl->nd; isf++) {
04255             los->sfeps[isf] = 1.0;
04256             if (ctl->nsf > 0) {
04257                 int isf = locate_irr(ctl->sfnu, ctl->nsf, ctl->nu[id]);
04258                 los->sfeps[isf] = LIN(ctl->sfnu[isf], atm->sfeps[isf],
04259                                     ctl->sfnu[isf + 1], atm->sfeps[isf + 1],
04260                                     ctl->nu[id]);
04261             }
04262         }
04263
04264         /* Leave raytracer... */
04265         break;

```

```

04266     }
04267
04268     /* Determine refractivity... */
04269     if (ctl->refrac && z <= zrefrac)
04270         n = 1 + refractivity(p, t);
04271     else
04272         n = 1;
04273
04274     /* Construct new tangent vector (first term)... */
04275     for (int i = 0; i < 3; i++)
04276         exl[i] = ex0[i] * n;
04277
04278     /* Compute gradient of refractivity... */
04279     if (ctl->refrac && z <= zrefrac) {
04280         for (int i = 0; i < 3; i++)
04281             xh[i] = x[i] + 0.5 * ds * ex0[i];
04282         cart2geo(xh, &z, &lon, &lat);
04283         intpol_atm(ctl, atm, z, &p, &t, q, k);
04284         n = refractivity(p, t);
04285         for (int i = 0; i < 3; i++) {
04286             xh[i] += h;
04287             cart2geo(xh, &z, &lon, &lat);
04288             intpol_atm(ctl, atm, z, &p, &t, q, k);
04289             ng[i] = (refractivity(p, t) - n) / h;
04290             xh[i] -= h;
04291         }
04292     } else
04293         for (int i = 0; i < 3; i++)
04294             ng[i] = 0;
04295
04296     /* Construct new tangent vector (second term)... */
04297     for (int i = 0; i < 3; i++)
04298         exl[i] += ds * ng[i];
04299
04300     /* Normalize new tangent vector... */
04301     norm = NORM(exl);
04302     for (int i = 0; i < 3; i++)
04303         exl[i] /= norm;
04304
04305     /* Determine next point of LOS... */
04306     for (int i = 0; i < 3; i++)
04307         x[i] += 0.5 * ds * (ex0[i] + exl[i]);
04308
04309     /* Copy tangent vector... */
04310     for (int i = 0; i < 3; i++)
04311         ex0[i] = exl[i];
04312 }
04313
04314 /* Get tangent point (to be done before changing segment lengths!)... */
04315 tangent_point(los, &obs->tpz[ir], &obs->tplon[ir], &obs->tplat[ir]);
04316
04317 /* Change segment lengths according to trapezoid rule... */
04318 for (int ip = los->np - 1; ip >= 1; ip--)
04319     los->ds[ip] = 0.5 * (los->ds[ip - 1] + los->ds[ip]);
04320 los->ds[0] *= 0.5;
04321
04322 /* Compute column density... */
04323 for (int ip = 0; ip < los->np; ip++)
04324     for (int ig = 0; ig < ctl->ng; ig++)
04325         los->u[ip][ig] = 10 * los->q[ip][ig] * los->p[ip]
04326             / (KB * los->t[ip]) * los->ds[ip];
04327 }
04328
04329 /*****
04330
04331 void read_atm(
04332     const char *dirname,
04333     const char *filename,
04334     ctl_t *ctl,
04335     atm_t *atm) {
04336
04337     FILE *in;
04338
04339     char file[LEN], line[LEN], *tok;
04340
04341     /* Init... */
04342     atm->np = 0;
04343
04344     /* Set filename... */
04345     if (dirname != NULL)
04346         sprintf(file, "%s/%s", dirname, filename);
04347     else
04348         sprintf(file, "%s", filename);
04349
04350     /* Write info... */
04351     LOG(1, "Read atmospheric data: %s", file);
04352

```

```

04353  /* Open file... */
04354  if (!(in = fopen(file, "r")))
04355      ERRMSG("Cannot open file!");
04356
04357  /* Read line... */
04358  while (fgets(line, LEN, in)) {
04359
04360      /* Read data... */
04361      TOK(line, tok, "%lg", atm->time[atm->np]);
04362      TOK(NULL, tok, "%lg", atm->z[atm->np]);
04363      TOK(NULL, tok, "%lg", atm->lon[atm->np]);
04364      TOK(NULL, tok, "%lg", atm->lat[atm->np]);
04365      TOK(NULL, tok, "%lg", atm->p[atm->np]);
04366      TOK(NULL, tok, "%lg", atm->t[atm->np]);
04367      for (int ig = 0; ig < ctl->ng; ig++)
04368          TOK(NULL, tok, "%lg", atm->q[ig][atm->np]);
04369      for (int iw = 0; iw < ctl->nw; iw++)
04370          TOK(NULL, tok, "%lg", atm->k[iw][atm->np]);
04371      if (ctl->ncl > 0 && atm->np == 0) {
04372          TOK(NULL, tok, "%lg", atm->clz);
04373          TOK(NULL, tok, "%lg", atm->cldz);
04374          for (int icl = 0; icl < ctl->ncl; icl++)
04375              TOK(NULL, tok, "%lg", atm->clk[icl]);
04376      }
04377      if (ctl->nsf > 0 && atm->np == 0) {
04378          TOK(NULL, tok, "%lg", atm->sfz);
04379          TOK(NULL, tok, "%lg", atm->sfp);
04380          TOK(NULL, tok, "%lg", atm->sft);
04381          for (int isf = 0; isf < ctl->nsf; isf++)
04382              TOK(NULL, tok, "%lg", atm->sfeps[isf]);
04383      }
04384
04385      /* Increment data point counter... */
04386      if ((++atm->np) > NP)
04387          ERRMSG("Too many data points!");
04388  }
04389
04390  /* Close file... */
04391  fclose(in);
04392
04393  /* Check number of points... */
04394  if (atm->np < 1)
04395      ERRMSG("Could not read any data!");
04396 }
04397
04398 /*****
04399
04400 void read_ctl(
04401     int argc,
04402     char *argv[],
04403     ctl_t * ctl) {
04404
04405     /* Write info... */
04406     LOG(1, "\nJuelich Rapid Spectral Simulation Code (JURASSIC)\n"
04407         "(executable: %s | version: %s | compiled: %s, %s)\n",
04408         argv[0], VERSION, __DATE__, __TIME__);
04409
04410     /* Emitters... */
04411     ctl->ng = (int) scan_ctl(argc, argv, "NG", -1, "0", NULL);
04412     if (ctl->ng < 0 || ctl->ng > NG)
04413         ERRMSG("Set 0 <= NG <= MAX!");
04414     for (int ig = 0; ig < ctl->ng; ig++)
04415         scan_ctl(argc, argv, "EMITTER", ig, "", ctl->emitter[ig]);
04416
04417     /* Radiance channels... */
04418     ctl->nd = (int) scan_ctl(argc, argv, "ND", -1, "0", NULL);
04419     if (ctl->nd < 0 || ctl->nd > ND)
04420         ERRMSG("Set 0 <= ND <= MAX!");
04421     for (int id = 0; id < ctl->nd; id++)
04422         ctl->nu[id] = scan_ctl(argc, argv, "NU", id, "", NULL);
04423
04424     /* Spectral windows... */
04425     ctl->nw = (int) scan_ctl(argc, argv, "NW", -1, "1", NULL);
04426     if (ctl->nw < 0 || ctl->nw > NW)
04427         ERRMSG("Set 0 <= NW <= MAX!");
04428     for (int id = 0; id < ctl->nd; id++)
04429         ctl->window[id] = (int) scan_ctl(argc, argv, "WINDOW", id, "0", NULL);
04430
04431     /* Cloud data... */
04432     ctl->ncl = (int) scan_ctl(argc, argv, "NCL", -1, "0", NULL);
04433     if (ctl->ncl < 0 || ctl->ncl > NCL)
04434         ERRMSG("Set 0 <= NCL <= MAX!");
04435     if (ctl->ncl == 1)
04436         ERRMSG("Set NCL > 1!");
04437     for (int icl = 0; icl < ctl->ncl; icl++)
04438         ctl->clnu[icl] = scan_ctl(argc, argv, "CLNU", icl, "", NULL);
04439

```

```

04440  /* Surface data... */
04441  ctl->nsf = (int) scan_ctl(argc, argv, "NSF", -1, "0", NULL);
04442  if (ctl->nsf < 0 || ctl->nsf > NSF)
04443      ERRMSG("Set 0 <= NSF <= MAX!");
04444  if (ctl->nsf == 1)
04445      ERRMSG("Set NSF > 1!");
04446  for (int isf = 0; isf < ctl->nsf; isf++)
04447      ctl->sfnu[isf] = scan_ctl(argc, argv, "SFNU", isf, "", NULL);
04448  ctl->sftype = (int) scan_ctl(argc, argv, "SFTYPE", -1, "2", NULL);
04449  if (ctl->sftype < 0 || ctl->sftype > 3)
04450      ERRMSG("Set 0 <= SFTYPE <= 3!");
04451  ctl->sfsza = scan_ctl(argc, argv, "SFSZA", -1, "-999", NULL);
04452
04453  /* Emissivity look-up tables... */
04454  scan_ctl(argc, argv, "TBLBASE", -1, "-", ctl->tblbase);
04455  ctl->tblfmt = (int) scan_ctl(argc, argv, "TBLFMT", -1, "1", NULL);
04456
04457  /* Hydrostatic equilibrium... */
04458  ctl->hydz = scan_ctl(argc, argv, "HYDZ", -1, "-999", NULL);
04459
04460  /* Continua... */
04461  ctl->ctm_co2 = (int) scan_ctl(argc, argv, "CTM_CO2", -1, "1", NULL);
04462  ctl->ctm_h2o = (int) scan_ctl(argc, argv, "CTM_H2O", -1, "1", NULL);
04463  ctl->ctm_n2 = (int) scan_ctl(argc, argv, "CTM_N2", -1, "1", NULL);
04464  ctl->ctm_o2 = (int) scan_ctl(argc, argv, "CTM_O2", -1, "1", NULL);
04465
04466  /* Ray-tracing... */
04467  ctl->refrac = (int) scan_ctl(argc, argv, "REFRAC", -1, "1", NULL);
04468  ctl->rayds = scan_ctl(argc, argv, "RAYDS", -1, "10", NULL);
04469  ctl->raydz = scan_ctl(argc, argv, "RAYDZ", -1, "0.1", NULL);
04470
04471  /* Field of view... */
04472  scan_ctl(argc, argv, "FOV", -1, "-", ctl->fov);
04473
04474  /* Retrieval interface... */
04475  ctl->retp_zmin = scan_ctl(argc, argv, "RETP_ZMIN", -1, "-999", NULL);
04476  ctl->retp_zmax = scan_ctl(argc, argv, "RETP_ZMAX", -1, "-999", NULL);
04477  ctl->rett_zmin = scan_ctl(argc, argv, "RETT_ZMIN", -1, "-999", NULL);
04478  ctl->rett_zmax = scan_ctl(argc, argv, "RETT_ZMAX", -1, "-999", NULL);
04479  for (int ig = 0; ig < ctl->ng; ig++) {
04480      ctl->retq_zmin[ig] = scan_ctl(argc, argv, "RETO_ZMIN", ig, "-999", NULL);
04481      ctl->retq_zmax[ig] = scan_ctl(argc, argv, "RETO_ZMAX", ig, "-999", NULL);
04482  }
04483  for (int iw = 0; iw < ctl->nw; iw++) {
04484      ctl->retk_zmin[iw] = scan_ctl(argc, argv, "RETK_ZMIN", iw, "-999", NULL);
04485      ctl->retk_zmax[iw] = scan_ctl(argc, argv, "RETK_ZMAX", iw, "-999", NULL);
04486  }
04487  ctl->ret_clz = (int) scan_ctl(argc, argv, "RET_CLZ", -1, "0", NULL);
04488  ctl->ret_cldz = (int) scan_ctl(argc, argv, "RET_CLDZ", -1, "0", NULL);
04489  ctl->ret_clk = (int) scan_ctl(argc, argv, "RET_CLK", -1, "0", NULL);
04490  ctl->ret_sfz = (int) scan_ctl(argc, argv, "RET_SFZ", -1, "0", NULL);
04491  ctl->ret_sfp = (int) scan_ctl(argc, argv, "RET_SFP", -1, "0", NULL);
04492  ctl->ret_sft = (int) scan_ctl(argc, argv, "RET_SFT", -1, "0", NULL);
04493  ctl->ret_sfeps = (int) scan_ctl(argc, argv, "RET_SFEPS", -1, "0", NULL);
04494
04495  /* Output flags... */
04496  ctl->write_bbt = (int) scan_ctl(argc, argv, "WRITE_BBT", -1, "0", NULL);
04497  ctl->write_matrix =
04498      (int) scan_ctl(argc, argv, "WRITE_MATRIX", -1, "0", NULL);
04499
04500  /* External forward models... */
04501  ctl->formod = (int) scan_ctl(argc, argv, "FORMOD", -1, "1", NULL);
04502  scan_ctl(argc, argv, "RFMBIN", -1, "-", ctl->rmbin);
04503  scan_ctl(argc, argv, "RFMHIT", -1, "-", ctl->rfmhit);
04504  for (int ig = 0; ig < ctl->ng; ig++)
04505      scan_ctl(argc, argv, "RFMXSC", ig, "-", ctl->rmtxsc[ig]);
04506 }
04507
04508 /*****
04509 void read_matrix(
04510     const char *dirname,
04511     const char *filename,
04512     gsl_matrix * matrix) {
04513
04514     FILE *in;
04515
04516     char dum[LEN], file[LEN], line[LEN];
04517
04518     double value;
04519
04520     int i, j;
04521
04522     /* Set filename... */
04523     if (dirname != NULL)
04524         sprintf(file, "%s/%s", dirname, filename);
04525     else

```

```

04527     sprintf(file, "%s", filename);
04528
04529     /* Write info... */
04530     LOG(1, "Read matrix: %s", file);
04531
04532     /* Open file... */
04533     if (!(in = fopen(file, "r")))
04534         ERRMSG("Cannot open file!");
04535
04536     /* Read data... */
04537     gsl_matrix_set_zero(matrix);
04538     while (fgets(line, LEN, in))
04539         if (sscanf(line, "%d %s %s %s %s %s %d %s %s %s %s %s %lg",
04540                 &i, dum, dum, dum, dum, dum,
04541                 &j, dum, dum, dum, dum, dum, &value) == 13)
04542         gsl_matrix_set(matrix, (size_t) i, (size_t) j, value);
04543
04544     /* Close file... */
04545     fclose(in);
04546 }
04547
04548 /*****
04549
04550 void read_obs(
04551     const char *dirname,
04552     const char *filename,
04553     ctl_t *ctl,
04554     obs_t *obs) {
04555
04556     FILE *in;
04557
04558     char file[LEN], line[LEN], *tok;
04559
04560     /* Init... */
04561     obs->nr = 0;
04562
04563     /* Set filename... */
04564     if (dirname != NULL)
04565         sprintf(file, "%s/%s", dirname, filename);
04566     else
04567         sprintf(file, "%s", filename);
04568
04569     /* Write info... */
04570     LOG(1, "Read observation data: %s", file);
04571
04572     /* Open file... */
04573     if (!(in = fopen(file, "r")))
04574         ERRMSG("Cannot open file!");
04575
04576     /* Read line... */
04577     while (fgets(line, LEN, in)) {
04578
04579         /* Read data... */
04580         TOK(line, tok, "%lg", obs->time[obs->nr]);
04581         TOK(NULL, tok, "%lg", obs->obsz[obs->nr]);
04582         TOK(NULL, tok, "%lg", obs->obslon[obs->nr]);
04583         TOK(NULL, tok, "%lg", obs->obslat[obs->nr]);
04584         TOK(NULL, tok, "%lg", obs->vpz[obs->nr]);
04585         TOK(NULL, tok, "%lg", obs->vplon[obs->nr]);
04586         TOK(NULL, tok, "%lg", obs->vplat[obs->nr]);
04587         TOK(NULL, tok, "%lg", obs->tpz[obs->nr]);
04588         TOK(NULL, tok, "%lg", obs->tplon[obs->nr]);
04589         TOK(NULL, tok, "%lg", obs->tplat[obs->nr]);
04590         for (int id = 0; id < ctl->nd; id++)
04591             TOK(NULL, tok, "%lg", obs->rad[id][obs->nr]);
04592         for (int id = 0; id < ctl->nd; id++)
04593             TOK(NULL, tok, "%lg", obs->tau[id][obs->nr]);
04594
04595         /* Increment counter... */
04596         if ((++obs->nr) > NR)
04597             ERRMSG("Too many rays!");
04598     }
04599
04600     /* Close file... */
04601     fclose(in);
04602
04603     /* Check number of points... */
04604     if (obs->nr < 1)
04605         ERRMSG("Could not read any data!");
04606 }
04607
04608 /*****
04609
04610 double read_obs_rfm(
04611     const char *basename,
04612     double z,
04613     double *nu,

```

```

04614 double *f,
04615 int n) {
04616
04617 FILE *in;
04618
04619 char filename[LEN];
04620
04621 double filt, fsum = 0, nu2[NSHAPE], *nurfm, *rad, radsum = 0;
04622
04623 int i, idx, ipt, npts;
04624
04625 /* Allocate... */
04626 ALLOC(nurfm, double,
04627       RFMNPTS);
04628 ALLOC(rad, double,
04629       RFMNPTS);
04630
04631 /* Search RFM spectrum... */
04632 sprintf(filename, "%s_%05d.asc", basename, (int) (z * 1000));
04633 if (!(in = fopen(filename, "r"))) {
04634     sprintf(filename, "%s_%05d.asc", basename, (int) (z * 1000) + 1);
04635     if (!(in = fopen(filename, "r")))
04636         ERRMSG("Cannot find RFM data file!");
04637 }
04638 fclose(in);
04639
04640 /* Read RFM spectrum... */
04641 read_rfm_spec(filename, nurfm, rad, &npts);
04642
04643 /* Set wavenumbers... */
04644 nu2[0] = nu[0];
04645 nu2[n - 1] = nu[n - 1];
04646 for (i = 1; i < n - 1; i++)
04647     nu2[i] = LIN(0.0, nu2[0], n - 1.0, nu2[n - 1], i);
04648
04649 /* Convolute... */
04650 for (ipt = 0; ipt < npts; ipt++)
04651     if (nurfm[ipt] >= nu2[0] && nurfm[ipt] <= nu2[n - 1]) {
04652         idx = locate_irr(nu2, n, nurfm[ipt]);
04653         filt = LIN(nu2[idx], f[idx], nu2[idx + 1], f[idx + 1], nurfm[ipt]);
04654         fsum += filt;
04655         radsum += filt * rad[ipt];
04656     }
04657
04658 /* Free... */
04659 free(nurfm);
04660 free(rad);
04661
04662 /* Return radiance... */
04663 return radsum / fsum;
04664 }
04665
04666 /*****
04667 void read_rfm_spec(
04668     const char *filename,
04669     double *nu,
04670     double *rad,
04671     int *npts) {
04672
04673     FILE *in;
04674
04675     char line[RFMLINE], *tok;
04676
04677     double dnu, nu0, nu1;
04678
04679     int i, ipt = 0;
04680
04681     /* Write info... */
04682     printf("Read RFM data: %s\n", filename);
04683
04684     /* Open file... */
04685     if (!(in = fopen(filename, "r")))
04686         ERRMSG("Cannot open file!");
04687
04688     /* Read header..... */
04689     for (i = 0; i < 4; i++)
04690         if (fgets(line, RFMLINE, in) == NULL)
04691             ERRMSG("Error while reading file header!");
04692         sscanf(line, "%d %lg %lg %lg", npts, &nu0, &dnu, &nu1);
04693     if (*npts > RFMNPTS)
04694         ERRMSG("Too many spectral grid points!");
04695
04696     /* Read radiance data... */
04697     while (fgets(line, RFMLINE, in) && ipt < *npts - 1) {
04698         if ((tok = strtok(line, " \t\n")) != NULL)
04699             if (sscanf(tok, "%lg", &rad[ipt]) == 1)
04700

```

```

04701     ipts++;
04702     while ((tok = strtok(NULL, " \t\n")) != NULL)
04703         if (sscanf(tok, "%lg", &rad[ipts]) == 1)
04704             ipts++;
04705 }
04706 if (ipts != *npts)
04707     ERRMSG("Error while reading RFM data!");
04708
04709 /* Compute wavenumbers... */
04710 for (ipts = 0; ipts < *npts; ipts++)
04711     nu[ipts] = LIN(0.0, nu0, (double) (*npts - 1), nu1, (double) ipts);
04712
04713 /* Close file... */
04714 fclose(in);
04715 }
04716
04717 /*****
04718
04719 void read_shape(
04720     const char *filename,
04721     double *x,
04722     double *y,
04723     int *n) {
04724
04725     FILE *in;
04726
04727     char line[LEN];
04728
04729     /* Write info... */
04730     LOG(1, "Read shape function: %s", filename);
04731
04732     /* Open file... */
04733     if (!(in = fopen(filename, "r")))
04734         ERRMSG("Cannot open file!");
04735
04736     /* Read data... */
04737     *n = 0;
04738     while (fgets(line, LEN, in))
04739         if (sscanf(line, "%lg %lg", &x[*n], &y[*n]) == 2)
04740             if (++(*n) > NSHAPE)
04741                 ERRMSG("Too many data points!");
04742
04743     /* Check number of points... */
04744     if (*n < 1)
04745         ERRMSG("Could not read any data!");
04746
04747     /* Close file... */
04748     fclose(in);
04749 }
04750
04751 /*****
04752
04753 void read_tbl(
04754     ctl_t *ctl,
04755     tbl_t *tbl) {
04756
04757     FILE *in;
04758
04759     char filename[2 * LEN], line[LEN];
04760
04761     double eps, press, temp, u;
04762
04763     /* Loop over trace gases and channels... */
04764     for (int id = 0; id < ctl->nd; id++)
04765         for (int ig = 0; ig < ctl->ng; ig++) {
04766
04767             /* Initialize... */
04768             tbl->np[id][ig] = -1;
04769             double eps_old = -999;
04770             double press_old = -999;
04771             double temp_old = -999;
04772             double u_old = -999;
04773
04774             /* Set filename... */
04775             sprintf(filename, "%s_%.4f_%.s.%s", ctl->tblbase,
04776                 ctl->nu[id], ctl->emitter[ig],
04777                 ctl->tblfmt == 1 ? "tab" : "bin");
04778
04779             /* Write info... */
04780             LOG(1, "Read emissivity table: %s", filename);
04781
04782             /* Try to open file... */
04783             if (!(in = fopen(filename, "r"))) {
04784                 WARN("Missing emissivity table: %s", filename);
04785                 continue;
04786             }
04787

```

```

04788     /* Read ASCII tables... */
04789     if (ctl->tblfmt == 1) {
04790
04791         /* Read data... */
04792         while (fgets(line, LEN, in)) {
04793
04794             /* Parse line... */
04795             if (sscanf(line, "%lg %lg %lg %lg", &press, &temp, &u, &eps) != 4)
04796                 continue;
04797
04798             /* Check ranges... */
04799             if (u < 0 || u > 1e30 || eps < 0 || eps > 1)
04800                 continue;
04801
04802             /* Determine pressure index... */
04803             if (press != press_old) {
04804                 press_old = press;
04805                 if ((++tbl->np[id][ig]) >= TBLNP)
04806                     ERRMSG("Too many pressure levels!");
04807                 tbl->nt[id][ig][tbl->np[id][ig]] = -1;
04808             }
04809
04810             /* Determine temperature index... */
04811             if (temp != temp_old) {
04812                 temp_old = temp;
04813                 if ((++tbl->nt[id][ig][tbl->np[id][ig]]) >= TBLNT)
04814                     ERRMSG("Too many temperatures!");
04815                 tbl->nu[id][ig][tbl->np[id][ig]]
04816                     [tbl->nt[id][ig][tbl->np[id][ig]]] = -1;
04817             }
04818
04819             /* Determine column density index... */
04820             if ((eps > eps_old && u > u_old) || tbl->nu[id][ig][tbl->np[id][ig]]
04821                 [tbl->nt[id][ig][tbl->np[id][ig]]] < 0) {
04822                 eps_old = eps;
04823                 u_old = u;
04824                 if ((++tbl->nu[id][ig][tbl->np[id][ig]]
04825                     [tbl->nt[id][ig][tbl->np[id][ig]]]) >= TBLNU) {
04826                     tbl->nu[id][ig][tbl->np[id][ig]]
04827                         [tbl->nt[id][ig][tbl->np[id][ig]]]--;
04828                     continue;
04829                 }
04830             }
04831
04832             /* Store data... */
04833             tbl->p[id][ig][tbl->np[id][ig]] = press;
04834             tbl->t[id][ig][tbl->np[id][ig]][tbl->nt[id][ig][tbl->np[id][ig]]]
04835                 = temp;
04836             tbl->u[id][ig][tbl->np[id][ig]][tbl->nt[id][ig][tbl->np[id][ig]]]
04837                 [tbl->nu[id][ig][tbl->np[id][ig]]]
04838                 [tbl->nt[id][ig][tbl->np[id][ig]]] = (float) u;
04839             tbl->eps[id][ig][tbl->np[id][ig]][tbl->nt[id][ig][tbl->np[id][ig]]]
04840                 [tbl->nu[id][ig][tbl->np[id][ig]]]
04841                 [tbl->nt[id][ig][tbl->np[id][ig]]] = (float) eps;
04842         }
04843
04844         /* Increment counters... */
04845         tbl->np[id][ig]++;
04846         for (int ip = 0; ip < tbl->np[id][ig]; ip++) {
04847             tbl->nt[id][ig][ip]++;
04848             for (int it = 0; it < tbl->nt[id][ig][ip]; it++)
04849                 tbl->nu[id][ig][ip][it]++;
04850         }
04851     }
04852
04853     /* Read binary data... */
04854     else if (ctl->tblfmt == 2) {
04855
04856         /* Read data... */
04857         FREAD(&tbl->np[id][ig], int,
04858             1,
04859             in);
04860         if (tbl->np[id][ig] > TBLNP)
04861             ERRMSG("Too many pressure levels!");
04862         FREAD(tbl->p[id][ig], double,
04863             (size_t) tbl->np[id][ig],
04864             in);
04865         for (int ip = 0; ip < tbl->np[id][ig]; ip++) {
04866             FREAD(&tbl->nt[id][ig][ip], int,
04867                 1,
04868                 in);
04869             if (tbl->nt[id][ig][ip] > TBLNT)
04870                 ERRMSG("Too many temperatures!");
04871             FREAD(tbl->t[id][ig][ip], double,
04872                 (size_t) tbl->nt[id][ig][ip],
04873                 in);
04874             for (int it = 0; it < tbl->nt[id][ig][ip]; it++) {

```



```

04875         FREAD(&tbl->nu[id][ig][ip][it], int,
04876                1,
04877                in);
04878         if (tbl->nu[id][ig][ip][it] > TBLNU)
04879             ERRMSG("Too many column densities!");
04880         FREAD(tbl->u[id][ig][ip][it], float,
04881               (size_t) tbl->nu[id][ig][ip][it],
04882               in);
04883         FREAD(tbl->eps[id][ig][ip][it], float,
04884               (size_t) tbl->nu[id][ig][ip][it],
04885               in);
04886     }
04887 }
04888 }
04889
04890 /* Error message... */
04891 else
04892     ERRMSG("Unknown look-up table format!");
04893
04894 /* Close file... */
04895 fclose(in);
04896 }
04897 }
04898
04899 /*****
04900
04901 double refractivity(
04902     double p,
04903     double t) {
04904
04905     /* Refractivity of air at 4 to 15 micron... */
04906     return 7.753e-05 * p / t;
04907 }
04908
04909 /*****
04910
04911 double scan_ctl(
04912     int argc,
04913     char *argv[],
04914     const char *varname,
04915     int arridx,
04916     const char *defvalue,
04917     char *value) {
04918
04919     FILE *in = NULL;
04920
04921     char dummy[LEN], fullname1[LEN], fullname2[LEN], line[LEN],
04922           rvarname[LEN], rval[LEN];
04923
04924     int contain = 0;
04925
04926     /* Open file... */
04927     if (argv[1][0] != '-')
04928         if (!(in = fopen(argv[1], "r")))
04929             ERRMSG("Cannot open file!");
04930
04931     /* Set full variable name... */
04932     if (arridx >= 0) {
04933         sprintf(fullname1, "%s[%d]", varname, arridx);
04934         sprintf(fullname2, "%s[*]", varname);
04935     } else {
04936         sprintf(fullname1, "%s", varname);
04937         sprintf(fullname2, "%s", varname);
04938     }
04939
04940     /* Read data... */
04941     if (in != NULL)
04942         while (fgets(line, LEN, in))
04943             if (sscanf(line, "%s %s %s", rvarname, dummy, rval) == 3)
04944                 if (strcasemp(rvarname, fullname1) == 0 ||
04945                     strcasemp(rvarname, fullname2) == 0) {
04946                     contain = 1;
04947                     break;
04948                 }
04949     for (int i = 1; i < argc - 1; i++)
04950         if (strcasemp(argv[i], fullname1) == 0 ||
04951             strcasemp(argv[i], fullname2) == 0) {
04952             sprintf(rval, "%s", argv[i + 1]);
04953             contain = 1;
04954             break;
04955         }
04956
04957     /* Close file... */
04958     if (in != NULL)
04959         fclose(in);
04960
04961     /* Check for missing variables... */

```

```

04962     if (!contain) {
04963         if (strlen(defvalue) > 0)
04964             sprintf(rval, "%s", defvalue);
04965         else
04966             ERRMSG("Missing variable %s!\n", fullname1);
04967     }
04968
04969     /* Write info... */
04970     LOG(1, "%s = %s", fullname1, rval);
04971
04972     /* Return values... */
04973     if (value != NULL)
04974         sprintf(value, "%s", rval);
04975     return atof(rval);
04976 }
04977
04978 /*****
04979
04980 double sza(
04981     double sec,
04982     double lon,
04983     double lat) {
04984
04985     /* Number of days and fraction with respect to 2000-01-01T12:00Z... */
04986     double D = sec / 86400 - 0.5;
04987
04988     /* Geocentric apparent ecliptic longitude [rad]... */
04989     double g = (357.529 + 0.98560028 * D) * M_PI / 180;
04990     double q = 280.459 + 0.98564736 * D;
04991     double L = (q + 1.915 * sin(g) + 0.020 * sin(2 * g)) * M_PI / 180;
04992
04993     /* Mean obliquity of the ecliptic [rad]... */
04994     double e = (23.439 - 0.00000036 * D) * M_PI / 180;
04995
04996     /* Declination [rad]... */
04997     double dec = asin(sin(e) * sin(L));
04998
04999     /* Right ascension [rad]... */
05000     double ra = atan2(cos(e) * sin(L), cos(L));
05001
05002     /* Greenwich Mean Sidereal Time [h]... */
05003     double GMST = 18.697374558 + 24.06570982441908 * D;
05004
05005     /* Local Sidereal Time [h]... */
05006     double LST = GMST + lon / 15;
05007
05008     /* Hour angle [rad]... */
05009     double h = LST / 12 * M_PI - ra;
05010
05011     /* Convert latitude... */
05012     lat *= M_PI / 180;
05013
05014     /* Return solar zenith angle [deg]... */
05015     return acos(sin(lat) * sin(dec) +
05016                cos(lat) * cos(dec) * cos(h)) * 180 / M_PI;
05017 }
05018
05019 /*****
05020
05021 void tangent_point(
05022     los_t * los,
05023     double *tpz,
05024     double *tplon,
05025     double *tplat) {
05026
05027     double dummy, v[3], v0[3], v2[3];
05028
05029     /* Find minimum altitude... */
05030     size_t ip = gsl_stats_min_index(los->z, 1, (size_t) los->np);
05031
05032     /* Nadir or zenith... */
05033     if (ip <= 0 || ip >= (size_t) los->np - 1) {
05034         *tpz = los->z[los->np - 1];
05035         *tplon = los->lon[los->np - 1];
05036         *tplat = los->lat[los->np - 1];
05037     }
05038
05039     /* Limb... */
05040     else {
05041
05042         /* Determine interpolating polynomial y=a*x^2+b*x+c... */
05043         double yy0 = los->z[ip - 1];
05044         double yy1 = los->z[ip];
05045         double yy2 = los->z[ip + 1];
05046         double x1 = sqrt(POW2(los->ds[ip]) - POW2(yy1 - yy0));
05047         double x2 = x1 + sqrt(POW2(los->ds[ip + 1]) - POW2(yy2 - yy1));
05048         double a = 1 / (x1 - x2) * (-(yy0 - yy1) / x1 + (yy0 - yy2) / x2);

```

```

05049     double b = -(yy0 - yy1) / x1 - a * x1;
05050     double c = yy0;
05051
05052     /* Get tangent point location... */
05053     double x = -b / (2 * a);
05054     *tpz = a * x * x + b * x + c;
05055     geo2cart(los->z[ip - 1], los->lon[ip - 1], los->lat[ip - 1], v0);
05056     geo2cart(los->z[ip + 1], los->lon[ip + 1], los->lat[ip + 1], v2);
05057     for (int i = 0; i < 3; i++)
05058         v[i] = LIN(0.0, v0[i], x2, v2[i], x);
05059     cart2geo(v, &dummy, tplon, tplat);
05060 }
05061 }
05062
05063 /*****
05064
05065 void time2jsec(
05066     int year,
05067     int mon,
05068     int day,
05069     int hour,
05070     int min,
05071     int sec,
05072     double remain,
05073     double *jsec) {
05074
05075     struct tm t0, t1;
05076
05077     t0.tm_year = 100;
05078     t0.tm_mon = 0;
05079     t0.tm_mday = 1;
05080     t0.tm_hour = 0;
05081     t0.tm_min = 0;
05082     t0.tm_sec = 0;
05083
05084     t1.tm_year = year - 1900;
05085     t1.tm_mon = mon - 1;
05086     t1.tm_mday = day;
05087     t1.tm_hour = hour;
05088     t1.tm_min = min;
05089     t1.tm_sec = sec;
05090
05091     *jsec = (double) timegm(&t1) - (double) timegm(&t0) + remain;
05092 }
05093
05094 /*****
05095
05096 void timer(
05097     const char *name,
05098     const char *file,
05099     const char *func,
05100     int line,
05101     int mode) {
05102
05103     static double w0[10];
05104
05105     static int l0[10], nt;
05106
05107     /* Start new timer... */
05108     if (mode == 1) {
05109         w0[nt] = omp_get_wtime();
05110         l0[nt] = line;
05111         if ((++nt) >= 10)
05112             ERRMSG("Too many timers!");
05113     }
05114
05115     /* Write elapsed time... */
05116     else {
05117
05118         /* Check timer index... */
05119         if (nt - 1 < 0)
05120             ERRMSG("Coding error!");
05121
05122         /* Write elapsed time... */
05123         LOG(1, "Timer '%s' (%s, %s, l%d-%d): %.3f sec",
05124             name, file, func, l0[nt - 1], line, omp_get_wtime() - w0[nt - 1]);
05125     }
05126
05127     /* Stop timer... */
05128     if (mode == 3)
05129         nt--;
05130 }
05131
05132 /*****
05133
05134 void write_atm(
05135     const char *dirname,

```

```

05136     const char *filename,
05137     ctl_t * ctl,
05138     atm_t * atm) {
05139
05140     FILE *out;
05141
05142     char file[LEN];
05143
05144     int n = 6;
05145
05146     /* Set filename... */
05147     if (dirname != NULL)
05148         sprintf(file, "%s/%s", dirname, filename);
05149     else
05150         sprintf(file, "%s", filename);
05151
05152     /* Write info... */
05153     LOG(1, "Write atmospheric data: %s", file);
05154
05155     /* Create file... */
05156     if (!(out = fopen(file, "w")))
05157         ERRMSG("Cannot create file!");
05158
05159     /* Write header... */
05160     fprintf(out,
05161             "# $1 = time (seconds since 2000-01-01T00:00Z)\n"
05162             "# $2 = altitude [km]\n"
05163             "# $3 = longitude [deg]\n"
05164             "# $4 = latitude [deg]\n"
05165             "# $5 = pressure [hPa]\n" "# $6 = temperature [K]\n");
05166     for (int ig = 0; ig < ctl->ng; ig++)
05167         fprintf(out, "# $5d = %s volume mixing ratio [ppv]\n",
05168                 ++n, ctl->emitter[ig]);
05169     for (int iw = 0; iw < ctl->nw; iw++)
05170         fprintf(out, "# $5d = extinction (window %d) [1/km]\n", ++n, iw);
05171     if (ctl->ncl > 0) {
05172         fprintf(out, "# $5d = cloud layer height [km]\n", ++n);
05173         fprintf(out, "# $5d = cloud layer depth [km]\n", ++n);
05174         for (int icl = 0; icl < ctl->ncl; icl++)
05175             fprintf(out, "# $5d = cloud layer extinction (%.4f cm^-1) [1/km]\n",
05176                     ++n, ctl->clnu[icl]);
05177     }
05178     if (ctl->nsf > 0) {
05179         fprintf(out, "# $5d = surface layer height [km]\n", ++n);
05180         fprintf(out, "# $5d = surface layer pressure [hPa]\n", ++n);
05181         fprintf(out, "# $5d = surface layer temperature [K]\n", ++n);
05182         for (int isf = 0; isf < ctl->nsf; isf++)
05183             fprintf(out, "# $5d = surface layer emissivity (%.4f cm^-1)\n",
05184                     ++n, ctl->sfnu[isf]);
05185     }
05186
05187     /* Write data... */
05188     for (int ip = 0; ip < atm->np; ip++) {
05189         if (ip == 0 || atm->time[ip] != atm->time[ip - 1])
05190             fprintf(out, "\n");
05191         fprintf(out, "%.2f %g %g %g %g", atm->time[ip], atm->z[ip],
05192                 atm->lon[ip], atm->lat[ip], atm->p[ip], atm->t[ip]);
05193         for (int ig = 0; ig < ctl->ng; ig++)
05194             fprintf(out, " %g", atm->q[ig][ip]);
05195         for (int iw = 0; iw < ctl->nw; iw++)
05196             fprintf(out, " %g", atm->k[iw][ip]);
05197         if (ctl->ncl > 0) {
05198             fprintf(out, " %g %g", atm->clz, atm->cldz);
05199             for (int icl = 0; icl < ctl->ncl; icl++)
05200                 fprintf(out, " %g", atm->clk[icl]);
05201         }
05202         if (ctl->nsf > 0) {
05203             fprintf(out, " %g %g %g", atm->sfz, atm->sfp, atm->sft);
05204             for (int isf = 0; isf < ctl->nsf; isf++)
05205                 fprintf(out, " %g", atm->sfeps[isf]);
05206         }
05207         fprintf(out, "\n");
05208     }
05209
05210     /* Close file... */
05211     fclose(out);
05212 }
05213
05214 /*****
05215
05216 void write_atm_rfm(
05217     const char *filename,
05218     ctl_t * ctl,
05219     atm_t * atm) {
05220
05221     FILE *out;
05222

```

```

05223 int ig, ip;
05224
05225 /* Write info... */
05226 printf("Write RFM data: %s\n", filename);
05227
05228 /* Create file... */
05229 if (!(out = fopen(filename, "w")))
05230     ERRMSG("Cannot create file!");
05231
05232 /* Write data... */
05233 fprintf(out, "%d\n", atm->np);
05234 fprintf(out, "*HGT [km]\n");
05235 for (ip = 0; ip < atm->np; ip++)
05236     fprintf(out, "%g\n", atm->z[ip]);
05237 fprintf(out, "*PRE [mb]\n");
05238 for (ip = 0; ip < atm->np; ip++)
05239     fprintf(out, "%g\n", atm->p[ip]);
05240 fprintf(out, "*TEM [K]\n");
05241 for (ip = 0; ip < atm->np; ip++)
05242     fprintf(out, "%g\n", atm->t[ip]);
05243 for (ig = 0; ig < ctl->ng; ig++) {
05244     fprintf(out, "%s [ppmv]\n", ctl->emitter[ig]);
05245     for (ip = 0; ip < atm->np; ip++)
05246         fprintf(out, "%g\n", atm->q[ig][ip] * 1e6);
05247 }
05248 fprintf(out, "*END\n");
05249
05250 /* Close file... */
05251 fclose(out);
05252 }
05253
05254 /*****
05255
05256 void write_matrix(
05257     const char *dirname,
05258     const char *filename,
05259     ctl_t *ctl,
05260     gsl_matrix *matrix,
05261     atm_t *atm,
05262     obs_t *obs,
05263     const char *rowsep,
05264     const char *colsep,
05265     const char *sort) {
05266
05267     FILE *out;
05268
05269     char file[LEN], quantity[LEN];
05270
05271     int *cida, *ciqa, *cipa, *cira, *rida, *riqa, *ripa, *rira;
05272
05273     size_t i, j, nc, nr;
05274
05275     /* Check output flag... */
05276     if (!(ctl->write_matrix))
05277         return;
05278
05279     /* Allocate... */
05280     ALLOC(cida, int,
05281         M);
05282     ALLOC(ciqa, int,
05283         N);
05284     ALLOC(cipa, int,
05285         N);
05286     ALLOC(cira, int,
05287         M);
05288     ALLOC(rida, int,
05289         M);
05290     ALLOC(riqa, int,
05291         N);
05292     ALLOC(ripa, int,
05293         N);
05294     ALLOC(rira, int,
05295         M);
05296
05297     /* Set filename... */
05298     if (dirname != NULL)
05299         sprintf(file, "%s/%s", dirname, filename);
05300     else
05301         sprintf(file, "%s", filename);
05302
05303     /* Write info... */
05304     LOG(1, "Write matrix: %s", file);
05305
05306     /* Create file... */
05307     if (!(out = fopen(file, "w")))
05308         ERRMSG("Cannot create file!");
05309

```

```

05310  /* Write header (row space)... */
05311  if (rowSpace[0] == 'y') {
05312
05313      fprintf(out,
05314              "# $1 = Row: index (measurement space)\n"
05315              "# $2 = Row: channel wavenumber [cm^-1]\n"
05316              "# $3 = Row: time (seconds since 2000-01-01T00:00Z)\n"
05317              "# $4 = Row: view point altitude [km]\n"
05318              "# $5 = Row: view point longitude [deg]\n"
05319              "# $6 = Row: view point latitude [deg]\n");
05320
05321      /* Get number of rows... */
05322      nr = obs2y(ctl, obs, NULL, rida, rira);
05323
05324  } else {
05325
05326      fprintf(out,
05327              "# $1 = Row: index (state space)\n"
05328              "# $2 = Row: name of quantity\n"
05329              "# $3 = Row: time (seconds since 2000-01-01T00:00Z)\n"
05330              "# $4 = Row: altitude [km]\n"
05331              "# $5 = Row: longitude [deg]\n" "# $6 = Row: latitude [deg]\n");
05332
05333      /* Get number of rows... */
05334      nr = atm2x(ctl, atm, NULL, rira, rira);
05335  }
05336
05337  /* Write header (column space)... */
05338  if (colSpace[0] == 'y') {
05339
05340      fprintf(out,
05341              "# $7 = Col: index (measurement space)\n"
05342              "# $8 = Col: channel wavenumber [cm^-1]\n"
05343              "# $9 = Col: time (seconds since 2000-01-01T00:00Z)\n"
05344              "# $10 = Col: view point altitude [km]\n"
05345              "# $11 = Col: view point longitude [deg]\n"
05346              "# $12 = Col: view point latitude [deg]\n");
05347
05348      /* Get number of columns... */
05349      nc = obs2y(ctl, obs, NULL, cida, cira);
05350
05351  } else {
05352
05353      fprintf(out,
05354              "# $7 = Col: index (state space)\n"
05355              "# $8 = Col: name of quantity\n"
05356              "# $9 = Col: time (seconds since 2000-01-01T00:00Z)\n"
05357              "# $10 = Col: altitude [km]\n"
05358              "# $11 = Col: longitude [deg]\n" "# $12 = Col: latitude [deg]\n");
05359
05360      /* Get number of columns... */
05361      nc = atm2x(ctl, atm, NULL, cira, cira);
05362  }
05363
05364  /* Write header entry... */
05365  fprintf(out, "# $13 = Matrix element\n\n");
05366
05367  /* Write matrix data... */
05368  i = j = 0;
05369  while (i < nr && j < nc) {
05370
05371      /* Write info about the row... */
05372      if (rowSpace[0] == 'y')
05373          fprintf(out, "%d %.4f %.2f %g %g %g",
05374                  (int) i, ctl->nu[rida[i]],
05375                  obs->time[rira[i]], obs->vpz[rira[i]],
05376                  obs->vplon[rira[i]], obs->vplat[rira[i]]);
05377      else {
05378          idx2name(ctl, rira[i], quantity);
05379          fprintf(out, "%d %s %.2f %g %g %g", (int) i, quantity,
05380                  atm->time[ripa[i]], atm->z[ripa[i]],
05381                  atm->lon[ripa[i]], atm->lat[ripa[i]]);
05382      }
05383
05384      /* Write info about the column... */
05385      if (colSpace[0] == 'y')
05386          fprintf(out, " %d %.4f %.2f %g %g %g",
05387                  (int) j, ctl->nu[cida[j]],
05388                  obs->time[cira[j]], obs->vpz[cira[j]],
05389                  obs->vplon[cira[j]], obs->vplat[cira[j]]);
05390      else {
05391          idx2name(ctl, cira[j], quantity);
05392          fprintf(out, " %d %s %.2f %g %g %g", (int) j, quantity,
05393                  atm->time[cipa[j]], atm->z[cipa[j]],
05394                  atm->lon[cipa[j]], atm->lat[cipa[j]]);
05395      }
05396  }

```

```

05397      /* Write matrix entry... */
05398      fprintf(out, " %g\n", gsl_matrix_get(matrix, i, j));
05399
05400      /* Set matrix indices... */
05401      if (sort[0] == 'r') {
05402          j++;
05403          if (j >= nc) {
05404              j = 0;
05405              i++;
05406              fprintf(out, "\n");
05407          }
05408      } else {
05409          i++;
05410          if (i >= nr) {
05411              i = 0;
05412              j++;
05413              fprintf(out, "\n");
05414          }
05415      }
05416  }
05417
05418  /* Close file... */
05419  fclose(out);
05420
05421  /* Free... */
05422  free(cida);
05423  free(ciga);
05424  free(cipa);
05425  free(cira);
05426  free(rida);
05427  free(riqa);
05428  free(ripa);
05429  free(rira);
05430 }
05431
05432 /*****
05433
05434 void write_obs(
05435     const char *dirname,
05436     const char *filename,
05437     ctl_t *ctl,
05438     obs_t *obs) {
05439
05440     FILE *out;
05441
05442     char file[LEN];
05443
05444     int n = 10;
05445
05446     /* Set filename... */
05447     if (dirname != NULL)
05448         sprintf(file, "%s/%s", dirname, filename);
05449     else
05450         sprintf(file, "%s", filename);
05451
05452     /* Write info... */
05453     LOG(1, "Write observation data: %s", file);
05454
05455     /* Create file... */
05456     if (!(out = fopen(file, "w")))
05457         ERRMSG("Cannot create file!");
05458
05459     /* Write header... */
05460     fprintf(out,
05461         "# $1 = time (seconds since 2000-01-01T00:00Z)\n"
05462         "# $2 = observer altitude [km]\n"
05463         "# $3 = observer longitude [deg]\n"
05464         "# $4 = observer latitude [deg]\n"
05465         "# $5 = view point altitude [km]\n"
05466         "# $6 = view point longitude [deg]\n"
05467         "# $7 = view point latitude [deg]\n"
05468         "# $8 = tangent point altitude [km]\n"
05469         "# $9 = tangent point longitude [deg]\n"
05470         "# $10 = tangent point latitude [deg]\n");
05471     for (int id = 0; id < ctl->nd; id++)
05472         fprintf(out, "# $%d = radiance (%.4f cm^-1) [W/(m^2 sr cm^-1)]\n",
05473             ++n, ctl->nu[id]);
05474     for (int id = 0; id < ctl->nd; id++)
05475         fprintf(out, "# $%d = transmittance (%.4f cm^-1) [-]\n", ++n,
05476             ctl->nu[id]);
05477
05478     /* Write data... */
05479     for (int ir = 0; ir < obs->nr; ir++) {
05480         if (ir == 0 || obs->time[ir] != obs->time[ir - 1])
05481             fprintf(out, "\n");
05482         fprintf(out, "%.2f %g %g %g %g %g %g %g %g", obs->time[ir],
05483             obs->obsz[ir], obs->obslon[ir], obs->obslat[ir],

```

```

05484         obs->vpz[ir], obs->vplon[ir], obs->vplat[ir],
05485         obs->tpz[ir], obs->tplon[ir], obs->tplat[ir]);
05486     for (int id = 0; id < ctl->nd; id++)
05487         fprintf(out, " %g", obs->rad[id][ir]);
05488     for (int id = 0; id < ctl->nd; id++)
05489         fprintf(out, " %g", obs->tau[id][ir]);
05490     fprintf(out, "\n");
05491 }
05492
05493 /* Close file... */
05494 fclose(out);
05495 }
05496
05497 /*****
05498
05499 void write_shape(
05500     const char *filename,
05501     double *x,
05502     double *y,
05503     int n) {
05504
05505     FILE *out;
05506
05507     /* Write info... */
05508     LOG(1, "Write shape function: %s", filename);
05509
05510     /* Create file... */
05511     if (!(out = fopen(filename, "w")))
05512         ERRMSG("Cannot create file!");
05513
05514     /* Write header... */
05515     fprintf(out,
05516             "# $1 = shape function x-value [-]\n"
05517             "# $2 = shape function y-value [-]\n\n");
05518
05519     /* Write data... */
05520     for (int i = 0; i < n; i++)
05521         fprintf(out, "%.10g %.10g\n", x[i], y[i]);
05522
05523     /* Close file... */
05524     fclose(out);
05525 }
05526
05527 /*****
05528
05529 void write_tbl(
05530     ctl_t *ctl,
05531     tbl_t *tbl) {
05532
05533     FILE *out;
05534
05535     char filename[2 * LEN];
05536
05537     /* Loop over emitters and detectors... */
05538     for (int ig = 0; ig < ctl->ng; ig++)
05539         for (int id = 0; id < ctl->nd; id++) {
05540
05541             /* Set filename... */
05542             sprintf(filename, "%s_%.4f_%.s.s", ctl->tblbase,
05543                     ctl->nu[id], ctl->emitter[ig],
05544                     ctl->tblfmt == 1 ? "tab" : "bin");
05545
05546             /* Write info... */
05547             LOG(1, "Write emissivity table: %s", filename);
05548
05549             /* Create file... */
05550             if (!(out = fopen(filename, "w")))
05551                 ERRMSG("Cannot create file!");
05552
05553             /* Write ASCII data... */
05554             if (ctl->tblfmt == 1) {
05555
05556                 /* Write header... */
05557                 fprintf(out,
05558                         "# $1 = pressure [hPa]\n"
05559                         "# $2 = temperature [K]\n"
05560                         "# $3 = column density [molecules/cm^2]\n"
05561                         "# $4 = emissivity [-]\n");
05562
05563                 /* Save table file... */
05564                 for (int ip = 0; ip < tbl->np[id][ig]; ip++)
05565                     for (int it = 0; it < tbl->nt[id][ig][ip]; it++) {
05566                         fprintf(out, "\n");
05567                         for (int iu = 0; iu < tbl->nu[id][ig][ip][it]; iu++)
05568                             fprintf(out, "%g %g %e %e\n",
05569                                     tbl->p[id][ig][ip], tbl->t[id][ig][ip][it],
05570                                     tbl->u[id][ig][ip][it][iu],

```



```

05571         tbl->eps[id][ig][ip][it][iu]);
05572     }
05573 }
05574
05575 /* Write binary data... */
05576 else if (ctl->tblfmt == 2) {
05577     FWRITE(&tbl->np[id][ig], int,
05578         1,
05579         out);
05580     FWRITE(tbl->p[id][ig], double,
05581         (size_t) tbl->np[id][ig],
05582         out);
05583     for (int ip = 0; ip < tbl->np[id][ig]; ip++) {
05584         FWRITE(&tbl->nt[id][ig][ip], int,
05585             1,
05586             out);
05587         FWRITE(tbl->t[id][ig][ip], double,
05588             (size_t) tbl->nt[id][ig][ip],
05589             out);
05590         for (int it = 0; it < tbl->nt[id][ig][ip]; it++) {
05591             FWRITE(&tbl->nu[id][ig][ip][it], int,
05592                 1,
05593                 out);
05594             FWRITE(tbl->u[id][ig][ip][it], float,
05595                 (size_t) tbl->nu[id][ig][ip][it],
05596                 out);
05597             FWRITE(tbl->eps[id][ig][ip][it], float,
05598                 (size_t) tbl->nu[id][ig][ip][it],
05599                 out);
05600         }
05601     }
05602 }
05603
05604 /* Error message... */
05605 else
05606     ERRMSG("Unknown look-up table format!");
05607
05608 /* Close file... */
05609 fclose(out);
05610 }
05611 }
05612
05613 /*****
05614
05615 void x2atm(
05616     ctl_t * ctl,
05617     gsl_vector * x,
05618     atm_t * atm) {
05619
05620     size_t n = 0;
05621
05622     /* Get pressure... */
05623     for (int ip = 0; ip < atm->np; ip++)
05624         if (atm->z[ip] >= ctl->retp_zmin && atm->z[ip] <= ctl->retp_zmax)
05625             x2atm_help(&atm->p[ip], x, &n);
05626
05627     /* Get temperature... */
05628     for (int ip = 0; ip < atm->np; ip++)
05629         if (atm->z[ip] >= ctl->rett_zmin && atm->z[ip] <= ctl->rett_zmax)
05630             x2atm_help(&atm->t[ip], x, &n);
05631
05632     /* Get volume mixing ratio... */
05633     for (int ig = 0; ig < ctl->ng; ig++)
05634         for (int ip = 0; ip < atm->np; ip++)
05635             if (atm->z[ip] >= ctl->retq_zmin[ig]
05636                 && atm->z[ip] <= ctl->retq_zmax[ig])
05637                 x2atm_help(&atm->q[ig][ip], x, &n);
05638
05639     /* Get extinction... */
05640     for (int iw = 0; iw < ctl->nw; iw++)
05641         for (int ip = 0; ip < atm->np; ip++)
05642             if (atm->z[ip] >= ctl->retk_zmin[iw]
05643                 && atm->z[ip] <= ctl->retk_zmax[iw])
05644                 x2atm_help(&atm->k[iw][ip], x, &n);
05645
05646     /* Get cloud data... */
05647     if (ctl->ret_clz)
05648         x2atm_help(&atm->clz, x, &n);
05649     if (ctl->ret_cldz)
05650         x2atm_help(&atm->cldz, x, &n);
05651     if (ctl->ret_clk)
05652         for (int icl = 0; icl < ctl->ncl; icl++)
05653             x2atm_help(&atm->clk[icl], x, &n);
05654
05655     /* Get surface data... */
05656     if (ctl->ret_sfz)
05657         x2atm_help(&atm->sfz, x, &n);

```

```

05658     if (ctl->ret_sfp)
05659         x2atm_help(&atm->sfp, x, &n);
05660     if (ctl->ret_sft)
05661         x2atm_help(&atm->sft, x, &n);
05662     if (ctl->ret_sfeps)
05663         for (int isf = 0; isf < ctl->nsf; isf++)
05664             x2atm_help(&atm->sfeps[isf], x, &n);
05665 }
05666
05667 /*****
05668
05669 void x2atm_help(
05670     double *value,
05671     gsl_vector * x,
05672     size_t *n) {
05673
05674     /* Get state vector element... */
05675     *value = gsl_vector_get(x, *n);
05676     (*n)++;
05677 }
05678
05679 /*****
05680
05681 void y2obs(
05682     ctl_t * ctl,
05683     gsl_vector * y,
05684     obs_t * obs) {
05685
05686     size_t m = 0;
05687
05688     /* Decompose measurement vector... */
05689     for (int ir = 0; ir < obs->nr; ir++)
05690         for (int id = 0; id < ctl->nd; id++)
05691             if (gsl_finite(obs->rad[id][ir])) {
05692                 obs->rad[id][ir] = gsl_vector_get(y, m);
05693                 m++;
05694             }
05695 }

```

## 5.19 jurassic.h File Reference

JURASSIC library declarations.

```

#include <gsl/gsl_math.h>
#include <gsl/gsl_blas.h>
#include <gsl/gsl_linalg.h>
#include <gsl/gsl_randist.h>
#include <gsl/gsl_rng.h>
#include <gsl/gsl_statistics.h>
#include <math.h>
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

```

### Data Structures

- struct [atm\\_t](#)  
*Atmospheric data.*
- struct [ctl\\_t](#)  
*Forward model control parameters.*
- struct [los\\_t](#)  
*Line-of-sight data.*
- struct [obs\\_t](#)  
*Observation geometry and radiance data.*
- struct [tbl\\_t](#)  
*Emissivity look-up tables.*

## Macros

- #define **ALLOC**(ptr, type, n)  
*Allocate memory.*
- #define **DIST**(a, b) sqrt(**DIST2**(a, b))  
*Compute Cartesian distance between two vectors.*
- #define **DIST2**(a, b) ((a[0]-b[0])\*(a[0]-b[0])+(a[1]-b[1])\*(a[1]-b[1])+(a[2]-b[2])\*(a[2]-b[2]))  
*Compute squared distance between two vectors.*
- #define **DOTP**(a, b) (a[0]\*b[0]+a[1]\*b[1]+a[2]\*b[2])  
*Compute dot product of two vectors.*
- #define **EXP**(x0, y0, x1, y1, x)  
*Compute exponential interpolation.*
- #define **FREAD**(ptr, type, size, out)  
*Read binary data.*
- #define **FWRITE**(ptr, type, size, out)  
*Write binary data.*
- #define **LIN**(x0, y0, x1, y1, x) ((y0)+((y1)-(y0))/((x1)-(x0))\*((x)-(x0)))  
*Compute linear interpolation.*
- #define **NORM**(a) sqrt(**DOTP**(a, a))  
*Compute norm of a vector.*
- #define **POW2**(x) ((x)\*(x))  
*Compute  $x^2$ .*
- #define **POW3**(x) ((x)\*(x)\*(x))  
*Compute  $x^3$ .*
- #define **TIMER**(name, mode) {timer(name, \_\_FILE\_\_, \_\_func\_\_, \_\_LINE\_\_, mode);}  
*Start or stop a timer.*
- #define **TOK**(line, tok, format, var)  
*Read string tokens.*
- #define **LOGLEV** 2  
*Level of log messages (0=none, 1=basic, 2=detailed, 3=debug).*
- #define **LOG**(level, ...)  
*Print log message.*
- #define **WARN**(...)  
*Print warning message.*
- #define **ERRMSG**(...)  
*Print error message and quit program.*
- #define **PRINT**(format, var)  
*Print macro for debugging.*
- #define **TMIN** 100.  
*Minimum temperature for source function [K].*
- #define **TMAX** 400.  
*Maximum temperature for source function [K].*
- #define **TSUN** 5780.  
*Effective temperature of the sun [K].*
- #define **C1** 1.19104259e-8  
*First spectroscopic constant ( $c_1 = 2 h c^2$ ) [ $W/(m^2 sr cm^{-4})$ ].*
- #define **C2** 1.43877506  
*Second spectroscopic constant ( $c_2 = h c / k$ ) [ $K/cm^{-1}$ ].*
- #define **G0** 9.80665  
*Standard gravity [ $m/s^2$ ].*
- #define **KB** 1.3806504e-23

- *Boltzmann constant [kg m<sup>2</sup>/(K s<sup>2</sup>)].*
- #define [NA](#) 6.02214199e23  
*Avogadro's number.*
- #define [HO](#) 7.0  
*Standard scale height [km].*
- #define [PO](#) 1013.25  
*Standard pressure [hPa].*
- #define [TO](#) 273.15  
*Standard temperature [K].*
- #define [RE](#) 6367.421  
*Mean radius of Earth [km].*
- #define [RI](#) 8.3144598  
*Ideal gas constant [J/(mol K)].*
- #define [ME](#) 5.976e24  
*Mass of Earth [kg].*
- #define [NCL](#) 8  
*Maximum number of cloud layer spectral grid points.*
- #define [ND](#) 64  
*Maximum number of radiance channels.*
- #define [NG](#) 8  
*Maximum number of emitters.*
- #define [NP](#) 256  
*Maximum number of atmospheric data points.*
- #define [NR](#) 256  
*Maximum number of ray paths.*
- #define [NSF](#) 8  
*Maximum number of surface layer spectral grid points.*
- #define [NW](#) 4  
*Maximum number of spectral windows.*
- #define [LEN](#) 10000  
*Maximum length of ASCII data lines.*
- #define [M](#) (NR\*ND)  
*Maximum size of measurement vector.*
- #define [N](#) ((2+NG+NW)\*NP+NCL+NSF+5)  
*Maximum size of state vector.*
- #define [NQ](#) (7+NG+NW+NCL+NSF)  
*Maximum number of quantities.*
- #define [NLOS](#) 4096  
*Maximum number of LOS points.*
- #define [NSHAPE](#) 10000  
*Maximum number of shape function grid points.*
- #define [NFOV](#) 5  
*Number of ray paths used for FOV calculations.*
- #define [TBLNP](#) 41  
*Maximum number of pressure levels in emissivity tables.*
- #define [TBLNT](#) 30  
*Maximum number of temperatures in emissivity tables.*
- #define [TBLNU](#) 320  
*Maximum number of column densities in emissivity tables.*
- #define [TBLNS](#) 1200  
*Maximum number of source function temperature levels.*

- `#define RFMNPTS 10000000`  
*Maximum number of RFM spectral grid points.*
- `#define RFMLINE 100000`  
*Maximum length of RFM data lines.*
- `#define IDXP 0`  
*Index for pressure.*
- `#define IDXT 1`  
*Index for temperature.*
- `#define IDXQ(ig) (2+ig)`  
*Indices for volume mixing ratios.*
- `#define ID XK(iw) (2+ctl->ng+iw)`  
*Indices for extinction.*
- `#define IDXCLZ (2+ctl->ng+ctl->nw)`  
*Index for cloud layer height.*
- `#define IDXCLDZ (3+ctl->ng+ctl->nw)`  
*Index for cloud layer depth.*
- `#define IDXCLK(icl) (4+ctl->ng+ctl->nw+icl)`  
*Indices for cloud layer extinction.*
- `#define IDXSfZ (4+ctl->ng+ctl->nw+ctl->ncl)`  
*Index for surface layer height.*
- `#define IDXSfP (5+ctl->ng+ctl->nw+ctl->ncl)`  
*Index for surface layer pressure.*
- `#define IDXSfT (6+ctl->ng+ctl->nw+ctl->ncl)`  
*Index for surface layer temperature.*
- `#define IDXSfEPS(isf) (7+ctl->ng+ctl->nw+ctl->ncl+isf)`  
*Indices for surface layer emissivity.*

## Functions

- `size_t atm2x (ctl_t *ctl, atm_t *atm, gsl_vector *x, int *iqa, int *ipa)`  
*Compose state vector or parameter vector.*
- `void atm2x_help (double value, int value_iqa, int value_ip, gsl_vector *x, int *iqa, int *ipa, size_t *n)`  
*Add element to state vector.*
- `double brightness (double rad, double nu)`  
*Compute brightness temperature.*
- `void cart2geo (double *x, double *z, double *lon, double *lat)`  
*Convert Cartesian coordinates to geolocation.*
- `void climatology (ctl_t *ctl, atm_t *atm_mean)`  
*Interpolate climatological data.*
- `double ctmcO2 (double nu, double p, double t, double u)`  
*Compute carbon dioxide continuum (optical depth).*
- `double ctmh2O (double nu, double p, double t, double q, double u)`  
*Compute water vapor continuum (optical depth).*
- `double ctmn2 (double nu, double p, double t)`  
*Compute nitrogen continuum (absorption coefficient).*
- `double ctmo2 (double nu, double p, double t)`  
*Compute oxygen continuum (absorption coefficient).*
- `void copy_atm (ctl_t *ctl, atm_t *atm_dest, atm_t *atm_src, int init)`  
*Copy and initialize atmospheric data.*
- `void copy_obs (ctl_t *ctl, obs_t *obs_dest, obs_t *obs_src, int init)`

- Copy and initialize observation data.*

  - int [find\\_emitter](#) ([ctl\\_t](#) \*ctl, const char \*emitter)

*Find index of an emitter.*
- void [formod](#) ([ctl\\_t](#) \*ctl, [atm\\_t](#) \*atm, [obs\\_t](#) \*obs)

*Determine ray paths and compute radiative transfer.*
- void [formod\\_continua](#) ([ctl\\_t](#) \*ctl, [los\\_t](#) \*los, int ip, double \*beta)

*Compute absorption coefficient of continua.*
- void [formod\\_fov](#) ([ctl\\_t](#) \*ctl, [obs\\_t](#) \*obs)

*Apply field of view convolution.*
- void [formod\\_pencil](#) ([ctl\\_t](#) \*ctl, [atm\\_t](#) \*atm, [obs\\_t](#) \*obs, int ir)

*Compute radiative transfer for a pencil beam.*
- void [formod\\_rfm](#) ([ctl\\_t](#) \*ctl, [atm\\_t](#) \*atm, [obs\\_t](#) \*obs)

*Apply RFM for radiative transfer calculations.*
- void [formod\\_srcfunc](#) ([ctl\\_t](#) \*ctl, [tbl\\_t](#) \*tbl, double t, double \*src)

*Compute Planck source function.*
- void [geo2cart](#) (double z, double lon, double lat, double \*x)

*Convert geolocation to Cartesian coordinates.*
- void [hydrostatic](#) ([ctl\\_t](#) \*ctl, [atm\\_t](#) \*atm)

*Set hydrostatic equilibrium.*
- void [idx2name](#) ([ctl\\_t](#) \*ctl, int idx, char \*quantity)

*Determine name of state vector quantity for given index.*
- void [init\\_srcfunc](#) ([ctl\\_t](#) \*ctl, [tbl\\_t](#) \*tbl)

*Initialize source function table.*
- void [intpol\\_atm](#) ([ctl\\_t](#) \*ctl, [atm\\_t](#) \*atm, double z, double \*p, double \*t, double \*q, double \*k)

*Interpolate atmospheric data.*
- void [intpol\\_tbl](#) ([ctl\\_t](#) \*ctl, [tbl\\_t](#) \*tbl, [los\\_t](#) \*los, int ip, double tau\_path[ND][NG], double tau\_seg[ND])

*Get transmittance from look-up tables.*
- double [intpol\\_tbl\\_eps](#) ([tbl\\_t](#) \*tbl, int ig, int id, int ip, int it, double u)

*Interpolate emissivity from look-up tables.*
- double [intpol\\_tbl\\_u](#) ([tbl\\_t](#) \*tbl, int ig, int id, int ip, int it, double eps)

*Interpolate column density from look-up tables.*
- void [jsec2time](#) (double jsec, int \*year, int \*mon, int \*day, int \*hour, int \*min, int \*sec, double \*remain)

*Convert seconds to date.*
- void [kernel](#) ([ctl\\_t](#) \*ctl, [atm\\_t](#) \*atm, [obs\\_t](#) \*obs, gsl\_matrix \*k)

*Compute Jacobians.*
- int [locate\\_irr](#) (double \*xx, int n, double x)

*Find array index for irregular grid.*
- int [locate\\_reg](#) (double \*xx, int n, double x)

*Find array index for regular grid.*
- int [locate\\_tbl](#) (float \*xx, int n, double x)

*Find array index in float array.*
- size\_t [obs2y](#) ([ctl\\_t](#) \*ctl, [obs\\_t](#) \*obs, gsl\_vector \*y, int \*ida, int \*ira)

*Compose measurement vector.*
- double [planck](#) (double t, double nu)

*Compute Planck function.*
- void [raytrace](#) ([ctl\\_t](#) \*ctl, [atm\\_t](#) \*atm, [obs\\_t](#) \*obs, [los\\_t](#) \*los, int ir)

*Do ray-tracing to determine LOS.*
- void [read\\_atm](#) (const char \*dirname, const char \*filename, [ctl\\_t](#) \*ctl, [atm\\_t](#) \*atm)

*Read atmospheric data.*
- void [read\\_ctl](#) (int argc, char \*argv[], [ctl\\_t](#) \*ctl)

*Read forward model control parameters.*

- void [read\\_matrix](#) (const char \*dirname, const char \*filename, gsl\_matrix \*matrix)  
*Read matrix.*
- void [read\\_obs](#) (const char \*dirname, const char \*filename, [ctl\\_t](#) \*ctl, [obs\\_t](#) \*obs)  
*Read observation data.*
- double [read\\_obs\\_rfm](#) (const char \*basename, double z, double \*nu, double \*f, int n)  
*Read observation data in RFM format.*
- void [read\\_rfm\\_spec](#) (const char \*filename, double \*nu, double \*rad, int \*npts)  
*Read RFM spectrum.*
- void [read\\_shape](#) (const char \*filename, double \*x, double \*y, int \*n)  
*Read shape function.*
- void [read\\_tbl](#) ([ctl\\_t](#) \*ctl, [tbl\\_t](#) \*tbl)  
*Read look-up table data.*
- double [refractivity](#) (double p, double t)  
*Compute refractivity (return value is  $n - 1$ ).*
- double [scan\\_ctl](#) (int argc, char \*argv[], const char \*varname, int arridx, const char \*defvalue, char \*value)  
*Search control parameter file for variable entry.*
- double [sza](#) (double sec, double lon, double lat)  
*Calculate solar zenith angle.*
- void [tangent\\_point](#) ([los\\_t](#) \*los, double \*tpz, double \*tplon, double \*tplat)  
*Find tangent point of a given LOS.*
- void [time2jsec](#) (int year, int mon, int day, int hour, int min, int sec, double remain, double \*jsec)  
*Convert date to seconds.*
- void [timer](#) (const char \*name, const char \*file, const char \*func, int line, int mode)  
*Measure wall-clock time.*
- void [write\\_atm](#) (const char \*dirname, const char \*filename, [ctl\\_t](#) \*ctl, [atm\\_t](#) \*atm)  
*Write atmospheric data.*
- void [write\\_atm\\_rfm](#) (const char \*filename, [ctl\\_t](#) \*ctl, [atm\\_t](#) \*atm)  
*Write atmospheric data in RFM format.*
- void [write\\_matrix](#) (const char \*dirname, const char \*filename, [ctl\\_t](#) \*ctl, gsl\_matrix \*matrix, [atm\\_t](#) \*atm, [obs\\_t](#) \*obs, const char \*rowsep, const char \*colsep, const char \*sort)  
*Write matrix.*
- void [write\\_obs](#) (const char \*dirname, const char \*filename, [ctl\\_t](#) \*ctl, [obs\\_t](#) \*obs)  
*Write observation data.*
- void [write\\_shape](#) (const char \*filename, double \*x, double \*y, int n)  
*Write shape function.*
- void [write\\_tbl](#) ([ctl\\_t](#) \*ctl, [tbl\\_t](#) \*tbl)  
*Write look-up table data.*
- void [x2atm](#) ([ctl\\_t](#) \*ctl, gsl\_vector \*x, [atm\\_t](#) \*atm)  
*Decompose parameter vector or state vector.*
- void [x2atm\\_help](#) (double \*value, gsl\_vector \*x, [size\\_t](#) \*n)  
*Get element from state vector.*
- void [y2obs](#) ([ctl\\_t](#) \*ctl, gsl\_vector \*y, [obs\\_t](#) \*obs)  
*Decompose measurement vector.*

### 5.19.1 Detailed Description

JURASSIC library declarations.

Definition in file [jurassic.h](#).

## 5.19.2 Macro Definition Documentation

### 5.19.2.1 ALLOC `#define ALLOC(`

```
ptr,  
type,  
n )
```

#### Value:

```
if((ptr=malloc((size_t)(n)*sizeof(type)))==NULL) \  
    ERRMSG("Out of memory!");
```

Allocate memory.

Definition at line 58 of file [jurassic.h](#).

### 5.19.2.2 DIST `#define DIST(`

```
a,  
b ) sqrt(DIST2(a, b))
```

Compute Cartesian distance between two vectors.

Definition at line 63 of file [jurassic.h](#).

### 5.19.2.3 DIST2 `#define DIST2(`

```
a,  
b ) ((a[0]-b[0])*(a[0]-b[0])+(a[1]-b[1])*(a[1]-b[1])+(a[2]-b[2])*(a[2]-b[2]))
```

Compute squared distance between two vectors.

Definition at line 66 of file [jurassic.h](#).

### 5.19.2.4 DOTP `#define DOTP(`

```
a,  
b ) (a[0]*b[0]+a[1]*b[1]+a[2]*b[2])
```

Compute dot product of two vectors.

Definition at line 70 of file [jurassic.h](#).



**5.19.2.5 EXP** `#define EXP(  
 x0,  
 y0,  
 x1,  
 y1,  
 x )`

**Value:**

```
((y0)>0 && (y1)>0)  
? ((y0)*exp(log((y1)/(y0)))/((x1)-(x0))*((x)-(x0))) \\  
: LIN(x0, y0, x1, y1, x) \
```

Compute exponential interpolation.

Definition at line 73 of file [jurassic.h](#).

**5.19.2.6 FREAD** `#define FREAD(  
 ptr,  
 type,  
 size,  
 out )`

**Value:**

```
{  
    if(fread(ptr, sizeof(type), size, out)!=size) \\  
        ERRMSG("Error while reading!"); \\  
}
```

Read binary data.

Definition at line 79 of file [jurassic.h](#).

**5.19.2.7 FWRITE** `#define FWRITE(  
 ptr,  
 type,  
 size,  
 out )`

**Value:**

```
{  
    if(fwrite(ptr, sizeof(type), size, out)!=size) \\  
        ERRMSG("Error while writing!"); \\  
}
```

Write binary data.

Definition at line 85 of file [jurassic.h](#).

**5.19.2.8 LIN** `#define LIN(  
 x0,  
 y0,  
 x1,  
 y1,  
 x ) ((y0) + ((y1) - (y0)) / ((x1) - (x0)) * ((x) - (x0)))`

Compute linear interpolation.

Definition at line 91 of file [jurassic.h](#).

**5.19.2.9 NORM** `#define NORM(  
 a ) sqrt(DOTP(a, a))`

Compute norm of a vector.

Definition at line 95 of file [jurassic.h](#).

**5.19.2.10 POW2** `#define POW2(  
 x ) ((x) * (x))`

Compute  $x^2$ .

Definition at line 98 of file [jurassic.h](#).

**5.19.2.11 POW3** `#define POW3(  
 x ) ((x) * (x) * (x))`

Compute  $x^3$ .

Definition at line 101 of file [jurassic.h](#).

**5.19.2.12 TIMER** `#define TIMER(  
 name,  
 mode ) {timer(name, __FILE__, __func__, __LINE__, mode);}`

Start or stop a timer.

Definition at line 104 of file [jurassic.h](#).

**5.19.2.13 TOK** `#define TOK(`  
`line,`  
`tok,`  
`format,`  
`var )`

**Value:**

```
{
    if((tok)=strtok((line), " \t")) {
        if(sscanf(tok, format, &(var))!=1) continue;
    } else ERRMSG("Error while reading!");
}
```

Read string tokens.

Definition at line 108 of file [jurassic.h](#).

**5.19.2.14 LOGLEV** `#define LOGLEV 2`

Level of log messages (0=none, 1=basic, 2=detailed, 3=debug).

Definition at line 120 of file [jurassic.h](#).

**5.19.2.15 LOG** `#define LOG(`  
`level,`  
`... )`

**Value:**

```
{
    if(level >= 2)
        printf(" ");
    if(level <= LOGLEV) {
        printf(__VA_ARGS__);
        printf("\n");
    }
}
```

Print log message.

Definition at line 124 of file [jurassic.h](#).

**5.19.2.16 WARN** `#define WARN(`  
`... )`

**Value:**

```
{
    printf("\nWarning (%s, %s, l%d): ", __FILE__, __func__, __LINE__);
    LOG(0, __VA_ARGS__);
}
```

Print warning message.

Definition at line 134 of file [jurassic.h](#).

**5.19.2.17 ERRMSG** `#define ERRMSG(  
    ... )`

**Value:**

```
{  
    printf("\nError (%s, %s, l%d): ", __FILE__, __func__, __LINE__);  
    LOG(0, __VA_ARGS__);  
    exit(EXIT_FAILURE);  
}
```

Print error message and quit program.

Definition at line 140 of file [jurassic.h](#).

**5.19.2.18 PRINT** `#define PRINT(  
    format,  
    var )`

**Value:**

```
printf("Print (%s, %s, l%d): %s= \"format\"\n",  
      __FILE__, __func__, __LINE__, #var, var);
```

Print macro for debugging.

Definition at line 147 of file [jurassic.h](#).

**5.19.2.19 TMIN** `#define TMIN 100.`

Minimum temperature for source function [K].

Definition at line 156 of file [jurassic.h](#).

**5.19.2.20 TMAX** `#define TMAX 400.`

Maximum temperature for source function [K].

Definition at line 159 of file [jurassic.h](#).

**5.19.2.21 TSUN** `#define TSUN 5780.`

Effective temperature of the sun [K].

Definition at line 162 of file [jurassic.h](#).

**5.19.2.22 C1** `#define C1 1.19104259e-8`

First spectroscopic constant ( $c_1 = 2 h c^2$ ) [ $\text{W}/(\text{m}^2 \text{ sr cm}^{-4})$ ].

Definition at line 165 of file [jurassic.h](#).

**5.19.2.23 C2** `#define C2 1.43877506`

Second spectroscopic constant ( $c_2 = h c / k$ ) [ $\text{K}/\text{cm}^{-1}$ ].

Definition at line 168 of file [jurassic.h](#).

**5.19.2.24 G0** `#define G0 9.80665`

Standard gravity [ $\text{m}/\text{s}^2$ ].

Definition at line 171 of file [jurassic.h](#).

**5.19.2.25 KB** `#define KB 1.3806504e-23`

Boltzmann constant [ $\text{kg m}^2/(\text{K s}^2)$ ].

Definition at line 174 of file [jurassic.h](#).

**5.19.2.26 NA** `#define NA 6.02214199e23`

Avogadro's number.

Definition at line 177 of file [jurassic.h](#).

**5.19.2.27 H0** `#define H0 7.0`

Standard scale height [km].

Definition at line 180 of file [jurassic.h](#).

**5.19.2.28 P0** `#define P0 1013.25`

Standard pressure [hPa].

Definition at line 183 of file [jurassic.h](#).

**5.19.2.29 T0** `#define T0 273.15`

Standard temperature [K].

Definition at line 186 of file [jurassic.h](#).

**5.19.2.30 RE** `#define RE 6367.421`

Mean radius of Earth [km].

Definition at line 189 of file [jurassic.h](#).

**5.19.2.31 RI** `#define RI 8.3144598`

Ideal gas constant [J/(mol K)].

Definition at line 192 of file [jurassic.h](#).

**5.19.2.32 ME** `#define ME 5.976e24`

Mass of Earth [kg].

Definition at line 195 of file [jurassic.h](#).

**5.19.2.33 NCL** `#define NCL 8`

Maximum number of cloud layer spectral grid points.

Definition at line 203 of file [jurassic.h](#).

**5.19.2.34 ND** `#define ND 64`

Maximum number of radiance channels.

Definition at line 208 of file [jurassic.h](#).

**5.19.2.35 NG** `#define NG 8`

Maximum number of emitters.

Definition at line 213 of file [jurassic.h](#).

**5.19.2.36 NP** `#define NP 256`

Maximum number of atmospheric data points.

Definition at line 218 of file [jurassic.h](#).

**5.19.2.37 NR** `#define NR 256`

Maximum number of ray paths.

Definition at line 223 of file [jurassic.h](#).

**5.19.2.38 NSF** `#define NSF 8`

Maximum number of surface layer spectral grid points.

Definition at line 228 of file [jurassic.h](#).

**5.19.2.39 NW** `#define NW 4`

Maximum number of spectral windows.

Definition at line 233 of file [jurassic.h](#).

**5.19.2.40 LEN** `#define LEN 10000`

Maximum length of ASCII data lines.

Definition at line 238 of file [jurassic.h](#).

**5.19.2.41 M** `#define M (NR*ND)`

Maximum size of measurement vector.

Definition at line 243 of file [jurassic.h](#).

**5.19.2.42 N** `#define N ((2+NG+NW)*NP+NCL+NSF+5)`

Maximum size of state vector.

Definition at line 248 of file [jurassic.h](#).

**5.19.2.43 NQ** `#define NQ (7+NG+NW+NCL+NSF)`

Maximum number of quantities.

Definition at line 253 of file [jurassic.h](#).

**5.19.2.44 NLOS** `#define NLOS 4096`

Maximum number of LOS points.

Definition at line 258 of file [jurassic.h](#).

**5.19.2.45 NSHAPE** `#define NSHAPE 10000`

Maximum number of shape function grid points.

Definition at line 263 of file [jurassic.h](#).



**5.19.2.46 NFOV** `#define NFOV 5`

Number of ray paths used for FOV calculations.

Definition at line 268 of file [jurassic.h](#).

**5.19.2.47 TBLNP** `#define TBLNP 41`

Maximum number of pressure levels in emissivity tables.

Definition at line 273 of file [jurassic.h](#).

**5.19.2.48 TBLNT** `#define TBLNT 30`

Maximum number of temperatures in emissivity tables.

Definition at line 278 of file [jurassic.h](#).

**5.19.2.49 TBLNU** `#define TBLNU 320`

Maximum number of column densities in emissivity tables.

Definition at line 283 of file [jurassic.h](#).

**5.19.2.50 TBLNS** `#define TBLNS 1200`

Maximum number of source function temperature levels.

Definition at line 288 of file [jurassic.h](#).

**5.19.2.51 RFMNPTS** `#define RFMNPTS 10000000`

Maximum number of RFM spectral grid points.

Definition at line 293 of file [jurassic.h](#).

**5.19.2.52 RFMLINE** `#define RFMLINE 100000`

Maximum length of RFM data lines.

Definition at line 298 of file [jurassic.h](#).

**5.19.2.53 IDXP** `#define IDXP 0`

Index for pressure.

Definition at line 306 of file [jurassic.h](#).

**5.19.2.54 IDXT** `#define IDXT 1`

Index for temperature.

Definition at line 309 of file [jurassic.h](#).

**5.19.2.55 IDXQ** `#define IDXQ(  
    ig ) (2+ig)`

Indices for volume mixing ratios.

Definition at line 312 of file [jurassic.h](#).

**5.19.2.56 IDXK** `#define IDXK(  
    iw ) (2+ctl->ng+iw)`

Indices for extinction.

Definition at line 315 of file [jurassic.h](#).

**5.19.2.57 IDXCLZ** `#define IDXCLZ (2+ctl->ng+ctl->nw)`

Index for cloud layer height.

Definition at line 318 of file [jurassic.h](#).

**5.19.2.58 IDXCLDZ** `#define IDXCLDZ (3+ctl->ng+ctl->nw)`

Index for cloud layer depth.

Definition at line 321 of file [jurassic.h](#).

**5.19.2.59 IDXCLK** `#define IDXCLK(  
 icl ) (4+ctl->ng+ctl->nw+icl)`

Indices for cloud layer extinction.

Definition at line 324 of file [jurassic.h](#).

**5.19.2.60 IDXSFZ** `#define IDXSFZ (4+ctl->ng+ctl->nw+ctl->ncl)`

Index for surface layer height.

Definition at line 327 of file [jurassic.h](#).

**5.19.2.61 IDXSFP** `#define IDXSFP (5+ctl->ng+ctl->nw+ctl->ncl)`

Index for surface layer pressure.

Definition at line 330 of file [jurassic.h](#).

**5.19.2.62 IDXSFT** `#define IDXSFT (6+ctl->ng+ctl->nw+ctl->ncl)`

Index for surface layer temperature.

Definition at line 333 of file [jurassic.h](#).

**5.19.2.63 IDXSFEPS** `#define IDXSFEPS(  
 isf ) (7+ctl->ng+ctl->nw+ctl->ncl+isf)`

Indices for surface layer emissivity.

Definition at line 336 of file [jurassic.h](#).

**5.19.3 Function Documentation**

```

5.19.3.1 atm2x() size_t atm2x (
    ctl_t * ctl,
    atm_t * atm,
    gsl_vector * x,
    int * iqa,
    int * ipa )

```

Compose state vector or parameter vector.

Definition at line 29 of file [jurassic.c](#).

```

00034     {
00035
00036     size_t n = 0;
00037
00038     /* Add pressure... */
00039     for (int ip = 0; ip < atm->np; ip++)
00040         if (atm->z[ip] >= ctl->retp_zmin && atm->z[ip] <= ctl->retp_zmax)
00041             atm2x_help(atm->p[ip], IDXP, ip, x, iqa, ipa, &n);
00042
00043     /* Add temperature... */
00044     for (int ip = 0; ip < atm->np; ip++)
00045         if (atm->z[ip] >= ctl->rett_zmin && atm->z[ip] <= ctl->rett_zmax)
00046             atm2x_help(atm->t[ip], IDXT, ip, x, iqa, ipa, &n);
00047
00048     /* Add volume mixing ratios... */
00049     for (int ig = 0; ig < ctl->ng; ig++)
00050         for (int ip = 0; ip < atm->np; ip++)
00051             if (atm->z[ip] >= ctl->retq_zmin[ig]
00052                 && atm->z[ip] <= ctl->retq_zmax[ig])
00053                 atm2x_help(atm->q[ig][ip], IDXQ(ig), ip, x, iqa, ipa, &n);
00054
00055     /* Add extinction... */
00056     for (int iw = 0; iw < ctl->nw; iw++)
00057         for (int ip = 0; ip < atm->np; ip++)
00058             if (atm->z[ip] >= ctl->retk_zmin[iw]
00059                 && atm->z[ip] <= ctl->retk_zmax[iw])
00060                 atm2x_help(atm->k[iw][ip], IDXK(iw), ip, x, iqa, ipa, &n);
00061
00062     /* Add cloud parameters... */
00063     if (ctl->ret_clz)
00064         atm2x_help(atm->clz, IDXCLZ, 0, x, iqa, ipa, &n);
00065     if (ctl->ret_cldz)
00066         atm2x_help(atm->cldz, IDXCLDZ, 0, x, iqa, ipa, &n);
00067     if (ctl->ret_clk)
00068         for (int icl = 0; icl < ctl->ncl; icl++)
00069             atm2x_help(atm->clk[icl], IDXCLK(icl), 0, x, iqa, ipa, &n);
00070
00071     /* Add surface parameters... */
00072     if (ctl->ret_sfz)
00073         atm2x_help(atm->sfz, IDXSFZ, 0, x, iqa, ipa, &n);
00074     if (ctl->ret_sfp)
00075         atm2x_help(atm->sfp, IDXSFZ, 0, x, iqa, ipa, &n);
00076     if (ctl->ret_sft)
00077         atm2x_help(atm->sft, IDXSFZ, 0, x, iqa, ipa, &n);
00078     if (ctl->ret_sfeps)
00079         for (int isf = 0; isf < ctl->nsf; isf++)
00080             atm2x_help(atm->sfeps[isf], IDXSFEPS(isf), 0, x, iqa, ipa, &n);
00081
00082     return n;
00083 }

```

Here is the call graph for this function:



**5.19.3.2 atm2x\_help()** void atm2x\_help (  
    double value,  
    int value\_iqua,  
    int value\_ip,  
    gsl\_vector \* x,  
    int \* iqua,  
    int \* ipa,  
    size\_t \* n )

Add element to state vector.

Definition at line 87 of file [jurassic.c](#).

```
00094     {  
00095  
00096     /* Add element to state vector... */  
00097     if (x != NULL)  
00098         gsl_vector_set(x, *n, value);  
00099     if (iqua != NULL)  
00100         iqua[*n] = value_iqua;  
00101     if (ipa != NULL)  
00102         ipa[*n] = value_ip;  
00103     (*n)++;  
00104 }
```

**5.19.3.3 brightness()** double brightness (  
    double rad,  
    double nu )

Compute brightness temperature.

Definition at line 109 of file [jurassic.c](#).

```
00111     {  
00112  
00113     return C2 * nu / gsl_log1p(C1 * POW3(nu) / rad);  
00114 }
```

**5.19.3.4 cart2geo()** void cart2geo (  
    double \* x,  
    double \* z,  
    double \* lon,  
    double \* lat )

Convert Cartesian coordinates to geolocation.

Definition at line 119 of file [jurassic.c](#).

```
00123     {  
00124  
00125     double radius = NORM(x);  
00126  
00127     *lat = asin(x[2] / radius) * 180 / M_PI;  
00128     *lon = atan2(x[1], x[0]) * 180 / M_PI;  
00129     *z = radius - RE;  
00130 }
```

```

5.19.3.5 climatology() void climatology (
    ctl_t * ctl,
    atm_t * atm_mean )

```

Interpolate climatological data.

Definition at line 134 of file [jurassic.c](#).

```

00136     {
00137
00138     static double z[121] = {
00139         0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
00140         20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37,
00141         38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55,
00142         56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73,
00143         74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91,
00144         92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107,
00145         108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120
00146     };
00147
00148     static double pre[121] = {
00149         1017, 901.083, 796.45, 702.227, 617.614, 541.644, 473.437, 412.288,
00150         357.603, 308.96, 265.994, 228.348, 195.619, 167.351, 143.039, 122.198,
00151         104.369, 89.141, 76.1528, 65.0804, 55.641, 47.591, 40.7233, 34.8637,
00152         29.8633, 25.5956, 21.9534, 18.8445, 16.1909, 13.9258, 11.9913,
00153         10.34, 8.92988, 7.72454, 6.6924, 5.80701, 5.04654, 4.39238, 3.82902,
00154         3.34337, 2.92413, 2.56128, 2.2464, 1.97258, 1.73384, 1.52519, 1.34242,
00155         1.18197, 1.04086, 0.916546, 0.806832, 0.709875, 0.624101, 0.548176,
00156         0.480974, 0.421507, 0.368904, 0.322408, 0.281386, 0.245249, 0.213465,
00157         0.185549, 0.161072, 0.139644, 0.120913, 0.104568, 0.0903249, 0.0779269,
00158         0.0671493, 0.0577962, 0.0496902, 0.0426736, 0.0366093, 0.0313743,
00159         0.0268598, 0.0229699, 0.0196206, 0.0167399, 0.0142646, 0.0121397,
00160         0.0103181, 0.00875775, 0.00742226, 0.00628076, 0.00530519, 0.00447183,
00161         0.00376124, 0.00315632, 0.00264248, 0.00220738, 0.00184003, 0.00153095,
00162         0.00127204, 0.00105608, 0.000876652, 0.00072798, 0.00060492,
00163         0.000503201, 0.000419226, 0.000349896, 0.000292659, 0.000245421,
00164         0.000206394, 0.000174125, 0.000147441, 0.000125333, 0.000106985,
00165         9.173e-05, 7.90172e-05, 6.84172e-05, 5.95574e-05, 5.21183e-05,
00166         4.58348e-05, 4.05127e-05, 3.59987e-05, 3.21583e-05, 2.88718e-05,
00167         2.60322e-05, 2.35687e-05, 2.14263e-05, 1.95489e-05
00168     };
00169
00170     static double tem[121] = {
00171         285.14, 279.34, 273.91, 268.3, 263.24, 256.55, 250.2, 242.82, 236.17,
00172         229.87, 225.04, 221.19, 218.85, 217.19, 216.2, 215.68, 215.42, 215.55,
00173         215.92, 216.4, 216.93, 217.45, 218, 218.68, 219.39, 220.25, 221.3,
00174         222.41, 223.88, 225.42, 227.2, 229.52, 231.89, 234.51, 236.85, 239.42,
00175         241.94, 244.57, 247.36, 250.32, 253.34, 255.82, 258.27, 260.39,
00176         262.03, 263.45, 264.2, 264.78, 264.67, 264.38, 263.24, 262.03, 260.02,
00177         258.09, 255.63, 253.28, 250.43, 247.81, 245.26, 242.77, 240.38,
00178         237.94, 235.79, 233.53, 231.5, 229.53, 227.6, 225.62, 223.77, 222.06,
00179         220.33, 218.69, 217.18, 215.64, 214.13, 212.52, 210.86, 209.25,
00180         207.49, 205.81, 204.11, 202.22, 200.32, 198.39, 195.92, 193.46,
00181         190.94, 188.31, 185.82, 183.57, 181.43, 179.74, 178.64, 178.1, 178.25,
00182         178.7, 179.41, 180.67, 182.31, 184.18, 186.6, 189.53, 192.66, 196.54,
00183         201.13, 205.93, 211.73, 217.86, 225, 233.53, 242.57, 252.14, 261.48,
00184         272.97, 285.26, 299.12, 312.2, 324.17, 338.34, 352.56, 365.28
00185     };
00186
00187     static double c2h2[121] = {
00188         1.352e-09, 2.83e-10, 1.269e-10, 6.926e-11, 4.346e-11, 2.909e-11,
00189         2.014e-11, 1.363e-11, 8.71e-12, 5.237e-12, 2.718e-12, 1.375e-12,
00190         5.786e-13, 2.16e-13, 7.317e-14, 2.551e-14, 1.055e-14, 4.758e-15,
00191         2.056e-15, 7.703e-16, 2.82e-16, 1.035e-16, 4.382e-17, 1.946e-17,
00192         9.638e-18, 5.2e-18, 2.811e-18, 1.494e-18, 7.925e-19, 4.213e-19,
00193         1.998e-19, 8.78e-20, 3.877e-20, 1.728e-20, 7.743e-21, 3.536e-21,
00194         1.623e-21, 7.508e-22, 3.508e-22, 1.65e-22, 7.837e-23, 3.733e-23,
00195         1.808e-23, 8.77e-24, 4.285e-24, 2.095e-24, 1.032e-24, 5.082e-25,
00196         2.506e-25, 1.236e-25, 6.088e-26, 2.996e-26, 1.465e-26, 0, 0, 0,
00197         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00198         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00199         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
00200     };
00201
00202     static double c2h6[121] = {
00203         2.667e-09, 2.02e-09, 1.658e-09, 1.404e-09, 1.234e-09, 1.109e-09,
00204         1.012e-09, 9.262e-10, 8.472e-10, 7.71e-10, 6.932e-10, 6.216e-10,
00205         5.503e-10, 4.87e-10, 4.342e-10, 3.861e-10, 3.347e-10, 2.772e-10,
00206         2.209e-10, 1.672e-10, 1.197e-10, 8.536e-11, 5.783e-11, 3.846e-11,
00207         2.495e-11, 1.592e-11, 1.017e-11, 6.327e-12, 3.895e-12, 2.403e-12,
00208         1.416e-12, 8.101e-13, 4.649e-13, 2.686e-13, 1.557e-13, 9.14e-14,
00209         5.386e-14, 3.19e-14, 1.903e-14, 1.14e-14, 6.875e-15, 4.154e-15,
00210         2.538e-15, 1.553e-15, 9.548e-16, 5.872e-16, 3.63e-16, 2.244e-16,
00211         1.388e-16, 8.587e-17, 5.308e-17, 3.279e-17, 2.017e-17, 1.238e-17,
00212         7.542e-18, 4.585e-18, 2.776e-18, 1.671e-18, 9.985e-19, 5.937e-19,

```

```
00213      3.518e-19, 2.07e-19, 1.215e-19, 7.06e-20, 4.097e-20, 2.37e-20,
00214      1.363e-20, 7.802e-21, 4.441e-21, 2.523e-21, 1.424e-21, 8.015e-22,
00215      4.497e-22, 2.505e-22, 1.391e-22, 7.691e-23, 4.238e-23, 2.331e-23,
00216      1.274e-23, 6.929e-24, 3.752e-24, 2.02e-24, 1.083e-24, 5.774e-25,
00217      3.041e-25, 1.593e-25, 8.308e-26, 4.299e-26, 2.195e-26, 1.112e-26,
00218      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00219      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00220  };
00221
00222  static double ccl4[121] = {
00223      1.075e-10, 1.075e-10, 1.075e-10, 1.075e-10, 1.075e-10, 1.075e-10,
00224      1.075e-10, 1.075e-10, 1.075e-10, 1.06e-10, 1.024e-10, 9.69e-11,
00225      8.93e-11, 8.078e-11, 7.213e-11, 6.307e-11, 5.383e-11, 4.49e-11,
00226      3.609e-11, 2.705e-11, 1.935e-11, 1.385e-11, 8.35e-12, 5.485e-12,
00227      3.853e-12, 2.22e-12, 5.875e-13, 3.445e-13, 1.015e-13, 6.075e-14,
00228      4.383e-14, 2.692e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00229      1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00230      1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00231      1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00232      1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00233      1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00234      1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00235      1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00236      1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14, 1e-14,
00237      1e-14, 1e-14, 1e-14
00238  };
00239
00240  static double ch4[121] = {
00241      1.864e-06, 1.835e-06, 1.819e-06, 1.805e-06, 1.796e-06, 1.788e-06,
00242      1.782e-06, 1.776e-06, 1.769e-06, 1.761e-06, 1.749e-06, 1.734e-06,
00243      1.716e-06, 1.692e-06, 1.654e-06, 1.61e-06, 1.567e-06, 1.502e-06,
00244      1.433e-06, 1.371e-06, 1.323e-06, 1.277e-06, 1.232e-06, 1.188e-06,
00245      1.147e-06, 1.108e-06, 1.07e-06, 1.027e-06, 9.854e-07, 9.416e-07,
00246      8.933e-07, 8.478e-07, 7.988e-07, 7.515e-07, 7.07e-07, 6.64e-07,
00247      6.239e-07, 5.864e-07, 5.512e-07, 5.184e-07, 4.87e-07, 4.571e-07,
00248      4.296e-07, 4.04e-07, 3.802e-07, 3.578e-07, 3.383e-07, 3.203e-07,
00249      3.032e-07, 2.889e-07, 2.76e-07, 2.635e-07, 2.519e-07, 2.409e-07,
00250      2.302e-07, 2.219e-07, 2.144e-07, 2.071e-07, 1.999e-07, 1.93e-07,
00251      1.862e-07, 1.795e-07, 1.731e-07, 1.668e-07, 1.607e-07, 1.548e-07,
00252      1.49e-07, 1.434e-07, 1.38e-07, 1.328e-07, 1.277e-07, 1.227e-07,
00253      1.18e-07, 1.134e-07, 1.089e-07, 1.046e-07, 1.004e-07, 9.635e-08,
00254      9.245e-08, 8.867e-08, 8.502e-08, 8.15e-08, 7.809e-08, 7.48e-08,
00255      7.159e-08, 6.849e-08, 6.55e-08, 6.262e-08, 5.98e-08, 5.708e-08,
00256      5.448e-08, 5.194e-08, 4.951e-08, 4.72e-08, 4.5e-08, 4.291e-08,
00257      4.093e-08, 3.905e-08, 3.729e-08, 3.563e-08, 3.408e-08, 3.265e-08,
00258      3.128e-08, 2.996e-08, 2.87e-08, 2.76e-08, 2.657e-08, 2.558e-08,
00259      2.467e-08, 2.385e-08, 2.307e-08, 2.234e-08, 2.168e-08, 2.108e-08,
00260      2.05e-08, 1.998e-08, 1.947e-08, 1.902e-08, 1.86e-08, 1.819e-08,
00261      1.782e-08
00262  };
00263
00264  static double clo[121] = {
00265      7.419e-15, 1.061e-14, 1.518e-14, 2.195e-14, 3.175e-14, 4.666e-14,
00266      6.872e-14, 1.03e-13, 1.553e-13, 2.375e-13, 3.664e-13, 5.684e-13,
00267      8.915e-13, 1.402e-12, 2.269e-12, 4.125e-12, 7.501e-12, 1.257e-11,
00268      2.048e-11, 3.338e-11, 5.44e-11, 8.846e-11, 1.008e-10, 1.082e-10,
00269      1.157e-10, 1.232e-10, 1.312e-10, 1.539e-10, 1.822e-10, 2.118e-10,
00270      2.387e-10, 2.687e-10, 2.875e-10, 3.031e-10, 3.23e-10, 3.648e-10,
00271      4.117e-10, 4.477e-10, 4.633e-10, 4.794e-10, 4.95e-10, 5.104e-10,
00272      5.259e-10, 5.062e-10, 4.742e-10, 4.443e-10, 4.051e-10, 3.659e-10,
00273      3.305e-10, 2.911e-10, 2.54e-10, 2.215e-10, 1.927e-10, 1.675e-10,
00274      1.452e-10, 1.259e-10, 1.09e-10, 9.416e-11, 8.119e-11, 6.991e-11,
00275      6.015e-11, 5.163e-11, 4.43e-11, 3.789e-11, 3.24e-11, 2.769e-11,
00276      2.361e-11, 2.011e-11, 1.71e-11, 1.453e-11, 1.233e-11, 1.045e-11,
00277      8.851e-12, 7.48e-12, 6.316e-12, 5.326e-12, 4.487e-12, 3.778e-12,
00278      3.176e-12, 2.665e-12, 2.234e-12, 1.87e-12, 1.563e-12, 1.304e-12,
00279      1.085e-12, 9.007e-13, 7.468e-13, 6.179e-13, 5.092e-13, 4.188e-13,
00280      3.442e-13, 2.816e-13, 2.304e-13, 1.885e-13, 1.542e-13, 1.263e-13,
00281      1.035e-13, 8.5e-14, 7.004e-14, 5.783e-14, 4.795e-14, 4.007e-14,
00282      3.345e-14, 2.792e-14, 2.33e-14, 1.978e-14, 1.686e-14, 1.438e-14,
00283      1.234e-14, 1.07e-14, 9.312e-15, 8.131e-15, 7.164e-15, 6.367e-15,
00284      5.67e-15, 5.088e-15, 4.565e-15, 4.138e-15, 3.769e-15, 3.432e-15,
00285      3.148e-15
00286  };
00287
00288  static double clono2[121] = {
00289      1.011e-13, 1.515e-13, 2.272e-13, 3.446e-13, 5.231e-13, 8.085e-13,
00290      1.253e-12, 1.979e-12, 3.149e-12, 5.092e-12, 8.312e-12, 1.366e-11,
00291      2.272e-11, 3.791e-11, 6.209e-11, 9.101e-11, 1.334e-10, 1.951e-10,
00292      2.853e-10, 3.94e-10, 4.771e-10, 5.771e-10, 6.675e-10, 7.665e-10,
00293      8.504e-10, 8.924e-10, 9.363e-10, 8.923e-10, 8.411e-10, 7.646e-10,
00294      6.525e-10, 5.576e-10, 4.398e-10, 3.403e-10, 2.612e-10, 1.915e-10,
00295      1.407e-10, 1.028e-10, 7.455e-11, 5.42e-11, 3.708e-11, 2.438e-11,
00296      1.618e-11, 1.075e-11, 7.17e-12, 4.784e-12, 3.205e-12, 2.147e-12,
00297      1.44e-12, 9.654e-13, 6.469e-13, 4.332e-13, 2.891e-13, 1.926e-13,
00298      1.274e-13, 8.422e-14, 5.547e-14, 3.636e-14, 2.368e-14, 1.536e-14,
00299      9.937e-15, 6.39e-15, 4.101e-15, 2.61e-15, 1.659e-15, 1.052e-15,
```

```
00300     6.638e-16, 4.172e-16, 2.61e-16, 1.63e-16, 1.013e-16, 6.275e-17,
00301     3.879e-17, 2.383e-17, 1.461e-17, 8.918e-18, 5.43e-18, 3.301e-18,
00302     1.997e-18, 1.203e-18, 7.216e-19, 4.311e-19, 2.564e-19, 1.519e-19,
00303     8.911e-20, 5.203e-20, 3.026e-20, 1.748e-20, 9.99e-21, 5.673e-21,
00304     3.215e-21, 1.799e-21, 1.006e-21, 5.628e-22, 3.146e-22, 1.766e-22,
00305     9.94e-23, 5.614e-23, 3.206e-23, 1.841e-23, 1.071e-23, 6.366e-24,
00306     3.776e-24, 2.238e-24, 1.326e-24, 8.253e-25, 5.201e-25, 3.279e-25,
00307     2.108e-25, 1.395e-25, 9.326e-26, 6.299e-26, 4.365e-26, 3.104e-26,
00308     2.219e-26, 1.621e-26, 1.185e-26, 8.92e-27, 6.804e-27, 5.191e-27,
00309     4.041e-27
00310 };
00311
00312 static double co[121] = {
00313     1.907e-07, 1.553e-07, 1.362e-07, 1.216e-07, 1.114e-07, 1.036e-07,
00314     9.737e-08, 9.152e-08, 8.559e-08, 7.966e-08, 7.277e-08, 6.615e-08,
00315     5.884e-08, 5.22e-08, 4.699e-08, 4.284e-08, 3.776e-08, 3.274e-08,
00316     2.845e-08, 2.479e-08, 2.246e-08, 2.054e-08, 1.991e-08, 1.951e-08,
00317     1.94e-08, 2.009e-08, 2.1e-08, 2.201e-08, 2.322e-08, 2.45e-08,
00318     2.602e-08, 2.73e-08, 2.867e-08, 2.998e-08, 3.135e-08, 3.255e-08,
00319     3.352e-08, 3.426e-08, 3.484e-08, 3.53e-08, 3.593e-08, 3.671e-08,
00320     3.759e-08, 3.945e-08, 4.192e-08, 4.49e-08, 5.03e-08, 5.703e-08,
00321     6.538e-08, 7.878e-08, 9.644e-08, 1.196e-07, 1.498e-07, 1.904e-07,
00322     2.422e-07, 3.055e-07, 3.804e-07, 4.747e-07, 5.899e-07, 7.272e-07,
00323     8.91e-07, 1.071e-06, 1.296e-06, 1.546e-06, 1.823e-06, 2.135e-06,
00324     2.44e-06, 2.714e-06, 2.967e-06, 3.189e-06, 3.391e-06, 3.58e-06,
00325     3.773e-06, 4.022e-06, 4.346e-06, 4.749e-06, 5.199e-06, 5.668e-06,
00326     6.157e-06, 6.688e-06, 7.254e-06, 7.867e-06, 8.539e-06, 9.26e-06,
00327     1.009e-05, 1.119e-05, 1.228e-05, 1.365e-05, 1.506e-05, 1.641e-05,
00328     1.784e-05, 1.952e-05, 2.132e-05, 2.323e-05, 2.531e-05, 2.754e-05,
00329     3.047e-05, 3.459e-05, 3.922e-05, 4.439e-05, 4.825e-05, 5.077e-05,
00330     5.34e-05, 5.618e-05, 5.909e-05, 6.207e-05, 6.519e-05, 6.845e-05,
00331     6.819e-05, 6.726e-05, 6.622e-05, 6.512e-05, 6.671e-05, 6.862e-05,
00332     7.048e-05, 7.264e-05, 7.3e-05, 7.3e-05, 7.3e-05, 7.3e-05, 7.3e-05
00333 };
00334
00335 static double cof2[121] = {
00336     7.5e-14, 1.055e-13, 1.485e-13, 2.111e-13, 3.001e-13, 4.333e-13,
00337     6.269e-13, 9.221e-13, 1.364e-12, 2.046e-12, 3.093e-12, 4.703e-12,
00338     7.225e-12, 1.113e-11, 1.66e-11, 2.088e-11, 2.626e-11, 3.433e-11,
00339     4.549e-11, 5.886e-11, 7.21e-11, 8.824e-11, 1.015e-10, 1.155e-10,
00340     1.288e-10, 1.388e-10, 1.497e-10, 1.554e-10, 1.606e-10, 1.639e-10,
00341     1.64e-10, 1.64e-10, 1.596e-10, 1.542e-10, 1.482e-10, 1.382e-10,
00342     1.289e-10, 1.198e-10, 1.109e-10, 1.026e-10, 9.484e-11, 8.75e-11,
00343     8.086e-11, 7.49e-11, 6.948e-11, 6.446e-11, 5.961e-11, 5.505e-11,
00344     5.085e-11, 4.586e-11, 4.1e-11, 3.665e-11, 3.235e-11, 2.842e-11,
00345     2.491e-11, 2.11e-11, 1.769e-11, 1.479e-11, 1.197e-11, 9.631e-12,
00346     7.74e-12, 6.201e-12, 4.963e-12, 3.956e-12, 3.151e-12, 2.507e-12,
00347     1.99e-12, 1.576e-12, 1.245e-12, 9.83e-13, 7.742e-13, 6.088e-13,
00348     4.782e-13, 3.745e-13, 2.929e-13, 2.286e-13, 1.782e-13, 1.388e-13,
00349     1.079e-13, 8.362e-14, 6.471e-14, 4.996e-14, 3.85e-14, 2.96e-14,
00350     2.265e-14, 1.729e-14, 1.317e-14, 9.998e-15, 7.549e-15, 5.683e-15,
00351     4.273e-15, 3.193e-15, 2.385e-15, 1.782e-15, 1.331e-15, 9.957e-16,
00352     7.461e-16, 5.601e-16, 4.228e-16, 3.201e-16, 2.438e-16, 1.878e-16,
00353     1.445e-16, 1.111e-16, 8.544e-17, 6.734e-17, 5.341e-17, 4.237e-17,
00354     3.394e-17, 2.759e-17, 2.254e-17, 1.851e-17, 1.54e-17, 1.297e-17,
00355     1.096e-17, 9.365e-18, 8e-18, 6.938e-18, 6.056e-18, 5.287e-18,
00356     4.662e-18
00357 };
00358
00359 static double f11[121] = {
00360     2.65e-10, 2.65e-10, 2.65e-10, 2.65e-10, 2.65e-10, 2.65e-10, 2.65e-10,
00361     2.65e-10, 2.65e-10, 2.65e-10, 2.65e-10, 2.65e-10, 2.635e-10, 2.536e-10,
00362     2.44e-10, 2.348e-10, 2.258e-10, 2.153e-10, 2.046e-10, 1.929e-10,
00363     1.782e-10, 1.648e-10, 1.463e-10, 1.291e-10, 1.1e-10, 8.874e-11,
00364     7.165e-11, 5.201e-11, 3.744e-11, 2.577e-11, 1.64e-11, 1.048e-11,
00365     5.993e-12, 3.345e-12, 1.839e-12, 9.264e-13, 4.688e-13, 2.329e-13,
00366     1.129e-13, 5.505e-14, 2.825e-14, 1.492e-14, 7.997e-15, 5.384e-15,
00367     3.988e-15, 2.955e-15, 2.196e-15, 1.632e-15, 1.214e-15, 9.025e-16,
00368     6.708e-16, 4.984e-16, 3.693e-16, 2.733e-16, 2.013e-16, 1.481e-16,
00369     1.087e-16, 7.945e-17, 5.782e-17, 4.195e-17, 3.038e-17, 2.19e-17,
00370     1.577e-17, 1.128e-17, 8.063e-18, 5.753e-18, 4.09e-18, 2.899e-18,
00371     2.048e-18, 1.444e-18, 1.015e-18, 7.12e-19, 4.985e-19, 3.474e-19,
00372     2.417e-19, 1.677e-19, 1.161e-19, 8.029e-20, 5.533e-20, 3.799e-20,
00373     2.602e-20, 1.776e-20, 1.209e-20, 8.202e-21, 5.522e-21, 3.707e-21,
00374     2.48e-21, 1.652e-21, 1.091e-21, 7.174e-22, 4.709e-22, 3.063e-22,
00375     1.991e-22, 1.294e-22, 8.412e-23, 5.483e-23, 3.581e-23, 2.345e-23,
00376     1.548e-23, 1.027e-23, 6.869e-24, 4.673e-24, 3.173e-24, 2.153e-24,
00377     1.461e-24, 1.028e-24, 7.302e-25, 5.188e-25, 3.739e-25, 2.753e-25,
00378     2.043e-25, 1.528e-25, 1.164e-25, 9.041e-26, 7.051e-26, 5.587e-26,
00379     4.428e-26, 3.588e-26, 2.936e-26, 2.402e-26, 1.995e-26
00380 };
00381
00382 static double f12[121] = {
00383     5.45e-10, 5.45e-10, 5.45e-10, 5.45e-10, 5.45e-10, 5.45e-10, 5.45e-10,
00384     5.45e-10, 5.45e-10, 5.45e-10, 5.45e-10, 5.45e-10, 5.429e-10, 5.291e-10,
00385     5.155e-10, 5.022e-10, 4.893e-10, 4.772e-10, 4.655e-10, 4.497e-10,
00386     4.249e-10, 4.015e-10, 3.632e-10, 3.261e-10, 2.858e-10, 2.408e-10,
```



```
00387      2.03e-10, 1.685e-10, 1.4e-10, 1.163e-10, 9.65e-11, 8.02e-11, 6.705e-11,
00388      5.624e-11, 4.764e-11, 4.249e-11, 3.792e-11, 3.315e-11, 2.819e-11,
00389      2.4e-11, 1.999e-11, 1.64e-11, 1.352e-11, 1.14e-11, 9.714e-12,
00390      8.28e-12, 7.176e-12, 6.251e-12, 5.446e-12, 4.72e-12, 4.081e-12,
00391      3.528e-12, 3.08e-12, 2.699e-12, 2.359e-12, 2.111e-12, 1.901e-12,
00392      1.709e-12, 1.534e-12, 1.376e-12, 1.233e-12, 1.103e-12, 9.869e-13,
00393      8.808e-13, 7.859e-13, 7.008e-13, 6.241e-13, 5.553e-13, 4.935e-13,
00394      4.383e-13, 3.889e-13, 3.447e-13, 3.054e-13, 2.702e-13, 2.389e-13,
00395      2.11e-13, 1.862e-13, 1.643e-13, 1.448e-13, 1.274e-13, 1.121e-13,
00396      9.844e-14, 8.638e-14, 7.572e-14, 6.62e-14, 5.782e-14, 5.045e-14,
00397      4.394e-14, 3.817e-14, 3.311e-14, 2.87e-14, 2.48e-14, 2.142e-14,
00398      1.851e-14, 1.599e-14, 1.383e-14, 1.196e-14, 1.036e-14, 9e-15,
00399      7.828e-15, 6.829e-15, 5.992e-15, 5.254e-15, 4.606e-15, 4.037e-15,
00400      3.583e-15, 3.19e-15, 2.841e-15, 2.542e-15, 2.291e-15, 2.07e-15,
00401      1.875e-15, 1.71e-15, 1.57e-15, 1.442e-15, 1.333e-15, 1.232e-15,
00402      1.147e-15, 1.071e-15, 1.001e-15, 9.396e-16
00403  };
00404
00405  static double f14[121] = {
00406      9e-11, 9e-11, 9e-11, 9e-11, 9e-11, 9e-11, 9e-11, 9e-11, 9e-11, 9e-11,
00407      9e-11, 9e-11, 9e-11, 9e-11, 9e-11, 8.91e-11, 8.73e-11, 8.46e-11,
00408      8.19e-11, 7.92e-11, 7.74e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00409      7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00410      7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00411      7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00412      7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00413      7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00414      7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00415      7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00416      7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00417      7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00418      7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00419      7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00420      7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00421      7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11,
00422      7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11, 7.65e-11
00423  };
00424
00425  static double f22[121] = {
00426      1.4e-10, 1.4e-10, 1.4e-10, 1.4e-10, 1.4e-10, 1.4e-10, 1.4e-10, 1.4e-10,
00427      1.4e-10, 1.4e-10, 1.4e-10, 1.372e-10, 1.317e-10, 1.235e-10, 1.153e-10,
00428      1.075e-10, 1.002e-10, 9.332e-11, 8.738e-11, 8.194e-11, 7.7e-11,
00429      7.165e-11, 6.753e-11, 6.341e-11, 5.971e-11, 5.6e-11, 5.229e-11,
00430      4.859e-11, 4.488e-11, 4.118e-11, 3.83e-11, 3.568e-11, 3.308e-11,
00431      3.047e-11, 2.82e-11, 2.594e-11, 2.409e-11, 2.237e-11, 2.065e-11,
00432      1.894e-11, 1.771e-11, 1.647e-11, 1.532e-11, 1.416e-11, 1.332e-11,
00433      1.246e-11, 1.161e-11, 1.087e-11, 1.017e-11, 9.471e-12, 8.853e-12,
00434      8.235e-12, 7.741e-12, 7.247e-12, 6.836e-12, 6.506e-12, 6.176e-12,
00435      5.913e-12, 5.65e-12, 5.419e-12, 5.221e-12, 5.024e-12, 4.859e-12,
00436      4.694e-12, 4.546e-12, 4.414e-12, 4.282e-12, 4.15e-12, 4.019e-12,
00437      3.903e-12, 3.805e-12, 3.706e-12, 3.607e-12, 3.508e-12, 3.41e-12,
00438      3.31e-12, 3.212e-12, 3.129e-12, 3.047e-12, 2.964e-12, 2.882e-12,
00439      2.8e-12, 2.734e-12, 2.668e-12, 2.602e-12, 2.537e-12, 2.471e-12,
00440      2.421e-12, 2.372e-12, 2.322e-12, 2.273e-12, 2.224e-12, 2.182e-12,
00441      2.141e-12, 2.1e-12, 2.059e-12, 2.018e-12, 1.977e-12, 1.935e-12,
00442      1.894e-12, 1.853e-12, 1.812e-12, 1.77e-12, 1.73e-12, 1.688e-12,
00443      1.647e-12, 1.606e-12, 1.565e-12, 1.524e-12, 1.483e-12, 1.441e-12,
00444      1.4e-12, 1.359e-12, 1.317e-12, 1.276e-12, 1.235e-12, 1.194e-12,
00445      1.153e-12, 1.112e-12, 1.071e-12, 1.029e-12, 9.883e-13
00446  };
00447
00448  static double h2o[121] = {
00449      0.01166, 0.008269, 0.005742, 0.003845, 0.00277, 0.001897, 0.001272,
00450      0.000827, 0.000539, 0.0003469, 0.0001579, 3.134e-05, 1.341e-05,
00451      6.764e-06, 4.498e-06, 3.703e-06, 3.724e-06, 3.899e-06, 4.002e-06,
00452      4.122e-06, 4.277e-06, 4.438e-06, 4.558e-06, 4.673e-06, 4.763e-06,
00453      4.809e-06, 4.856e-06, 4.936e-06, 5.021e-06, 5.114e-06, 5.222e-06,
00454      5.331e-06, 5.414e-06, 5.488e-06, 5.563e-06, 5.633e-06, 5.704e-06,
00455      5.767e-06, 5.819e-06, 5.872e-06, 5.914e-06, 5.949e-06, 5.984e-06,
00456      6.015e-06, 6.044e-06, 6.073e-06, 6.104e-06, 6.136e-06, 6.167e-06,
00457      6.189e-06, 6.208e-06, 6.226e-06, 6.212e-06, 6.185e-06, 6.158e-06,
00458      6.114e-06, 6.066e-06, 6.018e-06, 5.877e-06, 5.728e-06, 5.582e-06,
00459      5.437e-06, 5.296e-06, 5.156e-06, 5.02e-06, 4.886e-06, 4.754e-06,
00460      4.625e-06, 4.498e-06, 4.374e-06, 4.242e-06, 4.096e-06, 3.955e-06,
00461      3.817e-06, 3.683e-06, 3.491e-06, 3.204e-06, 2.94e-06, 2.696e-06,
00462      2.47e-06, 2.252e-06, 2.019e-06, 1.808e-06, 1.618e-06, 1.445e-06,
00463      1.285e-06, 1.105e-06, 9.489e-07, 8.121e-07, 6.938e-07, 5.924e-07,
00464      5.04e-07, 4.288e-07, 3.648e-07, 3.103e-07, 2.642e-07, 2.252e-07,
00465      1.921e-07, 1.643e-07, 1.408e-07, 1.211e-07, 1.048e-07, 9.063e-08,
00466      7.835e-08, 6.774e-08, 5.936e-08, 5.221e-08, 4.592e-08, 4.061e-08,
00467      3.62e-08, 3.236e-08, 2.902e-08, 2.62e-08, 2.383e-08, 2.171e-08,
00468      1.989e-08, 1.823e-08, 1.684e-08, 1.562e-08, 1.449e-08, 1.351e-08
00469  };
00470
00471  static double h2o2[121] = {
00472      1.779e-10, 7.938e-10, 8.953e-10, 8.032e-10, 6.564e-10, 5.159e-10,
00473      4.003e-10, 3.026e-10, 2.222e-10, 1.58e-10, 1.044e-10, 6.605e-11,
```

```
00474 3.413e-11, 1.453e-11, 1.062e-11, 1.009e-11, 9.597e-12, 1.175e-11,
00475 1.572e-11, 2.091e-11, 2.746e-11, 3.603e-11, 4.791e-11, 6.387e-11,
00476 8.239e-11, 1.007e-10, 1.23e-10, 1.363e-10, 1.489e-10, 1.585e-10,
00477 1.608e-10, 1.632e-10, 1.576e-10, 1.502e-10, 1.423e-10, 1.302e-10,
00478 1.192e-10, 1.085e-10, 9.795e-11, 8.854e-11, 8.057e-11, 7.36e-11,
00479 6.736e-11, 6.362e-11, 6.087e-11, 5.825e-11, 5.623e-11, 5.443e-11,
00480 5.27e-11, 5.098e-11, 4.931e-11, 4.769e-11, 4.611e-11, 4.458e-11,
00481 4.308e-11, 4.102e-11, 3.887e-11, 3.682e-11, 3.521e-11, 3.369e-11,
00482 3.224e-11, 3.082e-11, 2.946e-11, 2.814e-11, 2.687e-11, 2.566e-11,
00483 2.449e-11, 2.336e-11, 2.227e-11, 2.123e-11, 2.023e-11, 1.927e-11,
00484 1.835e-11, 1.746e-11, 1.661e-11, 1.58e-11, 1.502e-11, 1.428e-11,
00485 1.357e-11, 1.289e-11, 1.224e-11, 1.161e-11, 1.102e-11, 1.045e-11,
00486 9.895e-12, 9.369e-12, 8.866e-12, 8.386e-12, 7.922e-12, 7.479e-12,
00487 7.06e-12, 6.656e-12, 6.274e-12, 5.914e-12, 5.575e-12, 5.257e-12,
00488 4.959e-12, 4.679e-12, 4.42e-12, 4.178e-12, 3.954e-12, 3.75e-12,
00489 3.557e-12, 3.372e-12, 3.198e-12, 3.047e-12, 2.908e-12, 2.775e-12,
00490 2.653e-12, 2.544e-12, 2.442e-12, 2.346e-12, 2.26e-12, 2.183e-12,
00491 2.11e-12, 2.044e-12, 1.98e-12, 1.924e-12, 1.871e-12, 1.821e-12,
00492 1.775e-12
00493 };
00494
00495 static double hcn[121] = {
00496 5.5e-10, 5.5e-10, 5.5e-10, 5.5e-10, 5.5e-10, 5.5e-10, 5.5e-10,
00497 5.5e-10, 5.5e-10, 5.5e-10, 5.5e-10, 5.498e-10, 5.495e-10, 5.493e-10,
00498 5.49e-10, 5.488e-10, 4.717e-10, 3.946e-10, 3.174e-10, 2.4e-10,
00499 1.626e-10, 1.619e-10, 1.612e-10, 1.602e-10, 1.593e-10, 1.582e-10,
00500 1.572e-10, 1.56e-10, 1.549e-10, 1.539e-10, 1.53e-10, 1.519e-10,
00501 1.506e-10, 1.487e-10, 1.467e-10, 1.449e-10, 1.43e-10, 1.413e-10,
00502 1.397e-10, 1.382e-10, 1.368e-10, 1.354e-10, 1.337e-10, 1.315e-10,
00503 1.292e-10, 1.267e-10, 1.241e-10, 1.215e-10, 1.19e-10, 1.165e-10,
00504 1.141e-10, 1.118e-10, 1.096e-10, 1.072e-10, 1.047e-10, 1.021e-10,
00505 9.968e-11, 9.739e-11, 9.539e-11, 9.339e-11, 9.135e-11, 8.898e-11,
00506 8.664e-11, 8.439e-11, 8.249e-11, 8.075e-11, 7.904e-11, 7.735e-11,
00507 7.565e-11, 7.399e-11, 7.245e-11, 7.109e-11, 6.982e-11, 6.863e-11,
00508 6.755e-11, 6.657e-11, 6.587e-11, 6.527e-11, 6.476e-11, 6.428e-11,
00509 6.382e-11, 6.343e-11, 6.307e-11, 6.272e-11, 6.238e-11, 6.205e-11,
00510 6.17e-11, 6.137e-11, 6.102e-11, 6.072e-11, 6.046e-11, 6.03e-11,
00511 6.018e-11, 6.01e-11, 6.001e-11, 5.992e-11, 5.984e-11, 5.975e-11,
00512 5.967e-11, 5.958e-11, 5.95e-11, 5.941e-11, 5.933e-11, 5.925e-11,
00513 5.916e-11, 5.908e-11, 5.899e-11, 5.891e-11, 5.883e-11, 5.874e-11,
00514 5.866e-11, 5.858e-11, 5.85e-11, 5.841e-11, 5.833e-11, 5.825e-11,
00515 5.817e-11, 5.808e-11, 5.8e-11, 5.792e-11, 5.784e-11
00516 };
00517
00518 static double hno3[121] = {
00519 1.809e-10, 7.234e-10, 5.899e-10, 4.342e-10, 3.277e-10, 2.661e-10,
00520 2.35e-10, 2.267e-10, 2.389e-10, 2.651e-10, 3.255e-10, 4.099e-10,
00521 5.42e-10, 6.978e-10, 8.807e-10, 1.112e-09, 1.405e-09, 2.04e-09,
00522 3.111e-09, 4.5e-09, 5.762e-09, 7.37e-09, 7.852e-09, 8.109e-09,
00523 8.067e-09, 7.554e-09, 7.076e-09, 6.268e-09, 5.524e-09, 4.749e-09,
00524 3.909e-09, 3.223e-09, 2.517e-09, 1.942e-09, 1.493e-09, 1.122e-09,
00525 8.449e-10, 6.361e-10, 4.787e-10, 3.611e-10, 2.804e-10, 2.215e-10,
00526 1.758e-10, 1.441e-10, 1.197e-10, 9.953e-11, 8.505e-11, 7.334e-11,
00527 6.325e-11, 5.625e-11, 5.058e-11, 4.548e-11, 4.122e-11, 3.748e-11,
00528 3.402e-11, 3.088e-11, 2.8e-11, 2.536e-11, 2.293e-11, 2.072e-11,
00529 1.871e-11, 1.687e-11, 1.52e-11, 1.368e-11, 1.23e-11, 1.105e-11,
00530 9.922e-12, 8.898e-12, 7.972e-12, 7.139e-12, 6.385e-12, 5.708e-12,
00531 5.099e-12, 4.549e-12, 4.056e-12, 3.613e-12, 3.216e-12, 2.862e-12,
00532 2.544e-12, 2.259e-12, 2.004e-12, 1.776e-12, 1.572e-12, 1.391e-12,
00533 1.227e-12, 1.082e-12, 9.528e-13, 8.379e-13, 7.349e-13, 6.436e-13,
00534 5.634e-13, 4.917e-13, 4.291e-13, 3.745e-13, 3.267e-13, 2.854e-13,
00535 2.494e-13, 2.181e-13, 1.913e-13, 1.68e-13, 1.479e-13, 1.31e-13,
00536 1.159e-13, 1.025e-13, 9.067e-14, 8.113e-14, 7.281e-14, 6.535e-14,
00537 5.892e-14, 5.348e-14, 4.867e-14, 4.439e-14, 4.073e-14, 3.76e-14,
00538 3.476e-14, 3.229e-14, 3e-14, 2.807e-14, 2.635e-14, 2.473e-14,
00539 2.332e-14
00540 };
00541
00542 static double hno4[121] = {
00543 6.118e-12, 3.594e-12, 2.807e-12, 3.04e-12, 4.458e-12, 7.986e-12,
00544 1.509e-11, 2.661e-11, 3.738e-11, 4.652e-11, 4.429e-11, 3.992e-11,
00545 3.347e-11, 3.005e-11, 3.173e-11, 4.055e-11, 5.812e-11, 8.489e-11,
00546 1.19e-10, 1.482e-10, 1.766e-10, 2.103e-10, 2.35e-10, 2.598e-10,
00547 2.801e-10, 2.899e-10, 3e-10, 2.817e-10, 2.617e-10, 2.332e-10,
00548 1.933e-10, 1.605e-10, 1.232e-10, 9.285e-11, 6.941e-11, 4.951e-11,
00549 3.539e-11, 2.402e-11, 1.522e-11, 9.676e-12, 6.056e-12, 3.745e-12,
00550 2.34e-12, 1.463e-12, 9.186e-13, 5.769e-13, 3.322e-13, 1.853e-13,
00551 1.035e-13, 7.173e-14, 5.382e-14, 4.036e-14, 3.401e-14, 2.997e-14,
00552 2.635e-14, 2.316e-14, 2.034e-14, 1.783e-14, 1.56e-14, 1.363e-14,
00553 1.19e-14, 1.037e-14, 9.032e-15, 7.846e-15, 6.813e-15, 5.912e-15,
00554 5.121e-15, 4.431e-15, 3.829e-15, 3.306e-15, 2.851e-15, 2.456e-15,
00555 2.114e-15, 1.816e-15, 1.559e-15, 1.337e-15, 1.146e-15, 9.811e-16,
00556 8.389e-16, 7.162e-16, 6.109e-16, 5.203e-16, 4.425e-16, 3.76e-16,
00557 3.184e-16, 2.692e-16, 2.274e-16, 1.917e-16, 1.61e-16, 1.35e-16,
00558 1.131e-16, 9.437e-17, 7.874e-17, 6.57e-17, 5.481e-17, 4.579e-17,
00559 3.828e-17, 3.204e-17, 2.691e-17, 2.264e-17, 1.912e-17, 1.626e-17,
00560 1.382e-17, 1.174e-17, 9.972e-18, 8.603e-18, 7.45e-18, 6.453e-18,
```

```
00561      5.623e-18, 4.944e-18, 4.361e-18, 3.859e-18, 3.443e-18, 3.096e-18,
00562      2.788e-18, 2.528e-18, 2.293e-18, 2.099e-18, 1.929e-18, 1.773e-18,
00563      1.64e-18
00564  };
00565
00566  static double hoc1[121] = {
00567      1.056e-12, 1.194e-12, 1.35e-12, 1.531e-12, 1.737e-12, 1.982e-12,
00568      2.263e-12, 2.599e-12, 2.991e-12, 3.459e-12, 4.012e-12, 4.662e-12,
00569      5.438e-12, 6.35e-12, 7.425e-12, 8.686e-12, 1.016e-11, 1.188e-11,
00570      1.389e-11, 1.659e-11, 2.087e-11, 2.621e-11, 3.265e-11, 4.064e-11,
00571      4.859e-11, 5.441e-11, 6.09e-11, 6.373e-11, 6.611e-11, 6.94e-11,
00572      7.44e-11, 7.97e-11, 8.775e-11, 9.722e-11, 1.064e-10, 1.089e-10,
00573      1.114e-10, 1.106e-10, 1.053e-10, 1.004e-10, 9.006e-11, 7.778e-11,
00574      6.739e-11, 5.636e-11, 4.655e-11, 3.845e-11, 3.042e-11, 2.368e-11,
00575      1.845e-11, 1.442e-11, 1.127e-11, 8.814e-12, 6.544e-12, 4.763e-12,
00576      3.449e-12, 2.612e-12, 1.999e-12, 1.526e-12, 1.16e-12, 8.793e-13,
00577      6.655e-13, 5.017e-13, 3.778e-13, 2.829e-13, 2.117e-13, 1.582e-13,
00578      1.178e-13, 8.755e-14, 6.486e-14, 4.799e-14, 3.54e-14, 2.606e-14,
00579      1.916e-14, 1.403e-14, 1.026e-14, 7.48e-15, 5.446e-15, 3.961e-15,
00580      2.872e-15, 2.076e-15, 1.498e-15, 1.077e-15, 7.726e-16, 5.528e-16,
00581      3.929e-16, 2.785e-16, 1.969e-16, 1.386e-16, 9.69e-17, 6.747e-17,
00582      4.692e-17, 3.236e-17, 2.232e-17, 1.539e-17, 1.061e-17, 7.332e-18,
00583      5.076e-18, 3.522e-18, 2.461e-18, 1.726e-18, 1.22e-18, 8.75e-19,
00584      6.264e-19, 4.482e-19, 3.207e-19, 2.368e-19, 1.762e-19, 1.312e-19,
00585      9.891e-20, 7.595e-20, 5.87e-20, 4.567e-20, 3.612e-20, 2.904e-20,
00586      2.343e-20, 1.917e-20, 1.568e-20, 1.308e-20, 1.1e-20, 9.25e-21,
00587      7.881e-21
00588  };
00589
00590  static double n2o[121] = {
00591      3.17e-07, 3.17e-07, 3.17e-07, 3.17e-07, 3.17e-07, 3.17e-07, 3.17e-07,
00592      3.17e-07, 3.17e-07, 3.17e-07, 3.124e-07, 3.077e-07, 3.03e-07,
00593      2.984e-07, 2.938e-07, 2.892e-07, 2.847e-07, 2.779e-07, 2.705e-07,
00594      2.631e-07, 2.557e-07, 2.484e-07, 2.345e-07, 2.201e-07, 2.01e-07,
00595      1.754e-07, 1.532e-07, 1.329e-07, 1.154e-07, 1.003e-07, 8.735e-08,
00596      7.617e-08, 6.512e-08, 5.547e-08, 4.709e-08, 3.915e-08, 3.259e-08,
00597      2.738e-08, 2.327e-08, 1.98e-08, 1.711e-08, 1.493e-08, 1.306e-08,
00598      1.165e-08, 1.049e-08, 9.439e-09, 8.375e-09, 7.391e-09, 6.525e-09,
00599      5.759e-09, 5.083e-09, 4.485e-09, 3.953e-09, 3.601e-09, 3.27e-09,
00600      2.975e-09, 2.757e-09, 2.556e-09, 2.37e-09, 2.195e-09, 2.032e-09,
00601      1.912e-09, 1.79e-09, 1.679e-09, 1.572e-09, 1.482e-09, 1.402e-09,
00602      1.326e-09, 1.254e-09, 1.187e-09, 1.127e-09, 1.071e-09, 1.02e-09,
00603      9.673e-10, 9.193e-10, 8.752e-10, 8.379e-10, 8.017e-10, 7.66e-10,
00604      7.319e-10, 7.004e-10, 6.721e-10, 6.459e-10, 6.199e-10, 5.942e-10,
00605      5.703e-10, 5.488e-10, 5.283e-10, 5.082e-10, 4.877e-10, 4.696e-10,
00606      4.52e-10, 4.355e-10, 4.198e-10, 4.039e-10, 3.888e-10, 3.754e-10,
00607      3.624e-10, 3.499e-10, 3.381e-10, 3.267e-10, 3.163e-10, 3.058e-10,
00608      2.959e-10, 2.864e-10, 2.77e-10, 2.686e-10, 2.604e-10, 2.534e-10,
00609      2.462e-10, 2.386e-10, 2.318e-10, 2.247e-10, 2.189e-10, 2.133e-10,
00610      2.071e-10, 2.014e-10, 1.955e-10, 1.908e-10, 1.86e-10, 1.817e-10
00611  };
00612
00613  static double n2o5[121] = {
00614      1.231e-11, 3.035e-12, 1.702e-12, 9.877e-13, 8.081e-13, 9.039e-13,
00615      1.169e-12, 1.474e-12, 1.651e-12, 1.795e-12, 1.998e-12, 2.543e-12,
00616      4.398e-12, 7.698e-12, 1.28e-11, 2.131e-11, 3.548e-11, 5.894e-11,
00617      7.645e-11, 1.089e-10, 1.391e-10, 1.886e-10, 2.386e-10, 2.986e-10,
00618      3.487e-10, 3.994e-10, 4.5e-10, 4.6e-10, 4.591e-10, 4.1e-10, 3.488e-10,
00619      2.846e-10, 2.287e-10, 1.696e-10, 1.011e-10, 6.428e-11, 4.324e-11,
00620      2.225e-11, 6.214e-12, 3.608e-12, 8.793e-13, 4.491e-13, 1.04e-13,
00621      6.1e-14, 3.436e-14, 6.671e-15, 1.171e-15, 5.848e-16, 1.212e-16,
00622      1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00623      1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00624      1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00625      1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00626      1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00627      1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00628      1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16, 1e-16,
00629      1e-16, 1e-16
00630  };
00631
00632  static double nh3[121] = {
00633      1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10,
00634      1e-10, 1e-10, 1e-10, 9.444e-11, 8.488e-11, 7.241e-11, 5.785e-11,
00635      4.178e-11, 3.018e-11, 2.18e-11, 1.574e-11, 1.137e-11, 8.211e-12,
00636      5.973e-12, 4.327e-12, 3.118e-12, 2.234e-12, 1.573e-12, 1.04e-12,
00637      6.762e-13, 4.202e-13, 2.406e-13, 1.335e-13, 6.938e-14, 3.105e-14,
00638      1.609e-14, 1.033e-14, 6.432e-15, 4.031e-15, 2.555e-15, 1.656e-15,
00639      1.115e-15, 7.904e-16, 5.63e-16, 4.048e-16, 2.876e-16, 2.004e-16,
00640      1.356e-16, 9.237e-17, 6.235e-17, 4.223e-17, 3.009e-17, 2.328e-17,
00641      2.002e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00642      1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00643      1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00644      1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00645      1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00646      1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00647      1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
```

```
00648     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00649     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00650     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00651     1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17, 1.914e-17,
00652     1.914e-17
00653 };
00654
00655 static double no[121] = {
00656     2.586e-10, 4.143e-11, 1.566e-11, 9.591e-12, 8.088e-12, 8.462e-12,
00657     1.013e-11, 1.328e-11, 1.855e-11, 2.678e-11, 3.926e-11, 5.464e-11,
00658     7.012e-11, 8.912e-11, 1.127e-10, 1.347e-10, 1.498e-10, 1.544e-10,
00659     1.602e-10, 1.824e-10, 2.078e-10, 2.366e-10, 2.691e-10, 5.141e-10,
00660     8.259e-10, 1.254e-09, 1.849e-09, 2.473e-09, 3.294e-09, 4.16e-09,
00661     5.095e-09, 6.11e-09, 6.93e-09, 7.888e-09, 8.903e-09, 9.713e-09,
00662     1.052e-08, 1.115e-08, 1.173e-08, 1.21e-08, 1.228e-08, 1.239e-08,
00663     1.231e-08, 1.213e-08, 1.192e-08, 1.138e-08, 1.085e-08, 1.008e-08,
00664     9.224e-09, 8.389e-09, 7.262e-09, 6.278e-09, 5.335e-09, 4.388e-09,
00665     3.589e-09, 2.761e-09, 2.129e-09, 1.633e-09, 1.243e-09, 9.681e-10,
00666     8.355e-10, 7.665e-10, 7.442e-10, 8.584e-10, 9.732e-10, 1.063e-09,
00667     1.163e-09, 1.286e-09, 1.472e-09, 1.707e-09, 2.032e-09, 2.474e-09,
00668     2.977e-09, 3.506e-09, 4.102e-09, 5.013e-09, 6.493e-09, 8.414e-09,
00669     1.077e-08, 1.367e-08, 1.777e-08, 2.625e-08, 3.926e-08, 5.545e-08,
00670     7.195e-08, 9.464e-08, 1.404e-07, 2.183e-07, 3.329e-07, 4.535e-07,
00671     6.158e-07, 8.187e-07, 1.075e-06, 1.422e-06, 1.979e-06, 2.71e-06,
00672     3.58e-06, 4.573e-06, 5.951e-06, 7.999e-06, 1.072e-05, 1.372e-05,
00673     1.697e-05, 2.112e-05, 2.643e-05, 3.288e-05, 3.994e-05, 4.794e-05,
00674     5.606e-05, 6.383e-05, 7.286e-05, 8.156e-05, 8.883e-05, 9.469e-05,
00675     9.848e-05, 0.0001023, 0.0001066, 0.0001115, 0.0001145, 0.0001142,
00676     0.0001133
00677 };
00678
00679 static double no2[121] = {
00680     3.036e-09, 2.945e-10, 9.982e-11, 5.069e-11, 3.485e-11, 2.982e-11,
00681     2.947e-11, 3.164e-11, 3.714e-11, 4.586e-11, 6.164e-11, 8.041e-11,
00682     9.982e-11, 1.283e-10, 1.73e-10, 2.56e-10, 3.909e-10, 5.959e-10,
00683     9.081e-10, 1.384e-09, 1.788e-09, 2.189e-09, 2.686e-09, 3.091e-09,
00684     3.49e-09, 3.796e-09, 4.2e-09, 5.103e-09, 6.005e-09, 6.3e-09, 6.706e-09,
00685     7.07e-09, 7.434e-09, 7.663e-09, 7.788e-09, 7.8e-09, 7.597e-09,
00686     7.482e-09, 7.227e-09, 6.403e-09, 5.585e-09, 4.606e-09, 3.703e-09,
00687     2.984e-09, 2.183e-09, 1.48e-09, 8.441e-10, 5.994e-10, 3.799e-10,
00688     2.751e-10, 1.927e-10, 1.507e-10, 1.102e-10, 6.971e-11, 5.839e-11,
00689     3.904e-11, 3.087e-11, 2.176e-11, 1.464e-11, 1.209e-11, 8.497e-12,
00690     6.477e-12, 4.371e-12, 2.914e-12, 2.424e-12, 1.753e-12, 1.35e-12,
00691     9.417e-13, 6.622e-13, 5.148e-13, 3.841e-13, 3.446e-13, 3.01e-13,
00692     2.551e-13, 2.151e-13, 1.829e-13, 1.64e-13, 1.475e-13, 1.352e-13,
00693     1.155e-13, 9.963e-14, 9.771e-14, 9.577e-14, 9.384e-14, 9.186e-14,
00694     9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14,
00695     9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14,
00696     9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14,
00697     9e-14, 9e-14, 9e-14, 9e-14, 9e-14, 9e-14
00698 };
00699
00700 static double o3[121] = {
00701     2.218e-08, 3.394e-08, 3.869e-08, 4.219e-08, 4.501e-08, 4.778e-08,
00702     5.067e-08, 5.402e-08, 5.872e-08, 6.521e-08, 7.709e-08, 9.461e-08,
00703     1.269e-07, 1.853e-07, 2.723e-07, 3.964e-07, 5.773e-07, 8.2e-07,
00704     1.155e-06, 1.59e-06, 2.076e-06, 2.706e-06, 3.249e-06, 3.848e-06,
00705     4.459e-06, 4.986e-06, 5.573e-06, 5.958e-06, 6.328e-06, 6.661e-06,
00706     6.9e-06, 7.146e-06, 7.276e-06, 7.374e-06, 7.447e-06, 7.383e-06,
00707     7.321e-06, 7.161e-06, 6.879e-06, 6.611e-06, 6.216e-06, 5.765e-06,
00708     5.355e-06, 4.905e-06, 4.471e-06, 4.075e-06, 3.728e-06, 3.413e-06,
00709     3.125e-06, 2.856e-06, 2.607e-06, 2.379e-06, 2.17e-06, 1.978e-06,
00710     1.8e-06, 1.646e-06, 1.506e-06, 1.376e-06, 1.233e-06, 1.102e-06,
00711     9.839e-07, 8.771e-07, 7.814e-07, 6.947e-07, 6.102e-07, 5.228e-07,
00712     4.509e-07, 3.222e-07, 3.501e-07, 3.183e-07, 2.909e-07, 2.686e-07,
00713     2.476e-07, 2.984e-07, 2.109e-07, 2.003e-07, 2.013e-07, 2.022e-07,
00714     2.032e-07, 2.042e-07, 2.097e-07, 2.361e-07, 2.656e-07, 2.989e-07,
00715     3.37e-07, 3.826e-07, 4.489e-07, 5.26e-07, 6.189e-07, 7.312e-07,
00716     8.496e-07, 8.444e-07, 8.392e-07, 8.339e-07, 8.286e-07, 8.234e-07,
00717     8.181e-07, 8.129e-07, 8.077e-07, 8.026e-07, 6.918e-07, 5.176e-07,
00718     3.865e-07, 2.885e-07, 2.156e-07, 1.619e-07, 1.219e-07, 9.161e-08,
00719     6.972e-08, 5.399e-08, 3.498e-08, 2.111e-08, 1.322e-08, 8.482e-09,
00720     5.527e-09, 3.423e-09, 2.071e-09, 1.314e-09, 8.529e-10, 5.503e-10,
00721     3.665e-10
00722 };
00723
00724 static double ocs[121] = {
00725     6e-10, 6e-10, 6e-10, 6e-10, 6e-10, 6e-10, 6e-10, 6e-10, 6e-10, 5.997e-10,
00726     5.989e-10, 5.881e-10, 5.765e-10, 5.433e-10, 5.074e-10, 4.567e-10,
00727     4.067e-10, 3.601e-10, 3.093e-10, 2.619e-10, 2.232e-10, 1.805e-10,
00728     1.46e-10, 1.187e-10, 8.03e-11, 5.435e-11, 3.686e-11, 2.217e-11,
00729     1.341e-11, 8.756e-12, 4.511e-12, 2.37e-12, 1.264e-12, 8.28e-13,
00730     5.263e-13, 3.209e-13, 1.717e-13, 9.068e-14, 4.709e-14, 2.389e-14,
00731     1.236e-14, 1.127e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00732     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00733     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00734     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
```

```

00735     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00736     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00737     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00738     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00739     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00740     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00741     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00742     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00743     1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14, 1.091e-14,
00744     1.091e-14, 1.091e-14, 1.091e-14
00745 };
00746
00747 static double sf6[121] = {
00748     4.103e-12, 4.103e-12, 4.103e-12, 4.103e-12, 4.103e-12, 4.103e-12,
00749     4.103e-12, 4.103e-12, 4.103e-12, 4.087e-12, 4.064e-12, 4.023e-12,
00750     3.988e-12, 3.941e-12, 3.884e-12, 3.755e-12, 3.622e-12, 3.484e-12,
00751     3.32e-12, 3.144e-12, 2.978e-12, 2.811e-12, 2.653e-12, 2.489e-12,
00752     2.332e-12, 2.199e-12, 2.089e-12, 2.013e-12, 1.953e-12, 1.898e-12,
00753     1.859e-12, 1.826e-12, 1.798e-12, 1.776e-12, 1.757e-12, 1.742e-12,
00754     1.728e-12, 1.717e-12, 1.707e-12, 1.698e-12, 1.691e-12, 1.685e-12,
00755     1.679e-12, 1.675e-12, 1.671e-12, 1.668e-12, 1.665e-12, 1.663e-12,
00756     1.661e-12, 1.659e-12, 1.658e-12, 1.657e-12, 1.656e-12, 1.655e-12,
00757     1.654e-12, 1.653e-12, 1.653e-12, 1.652e-12, 1.652e-12, 1.652e-12,
00758     1.651e-12, 1.651e-12, 1.651e-12, 1.651e-12, 1.651e-12, 1.651e-12,
00759     1.651e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00760     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00761     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00762     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00763     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00764     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00765     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12,
00766     1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12, 1.65e-12
00767 };
00768
00769 static double so2[121] = {
00770     1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10,
00771     1e-10, 1e-10, 9.867e-11, 9.537e-11, 9e-11, 8.404e-11, 7.799e-11,
00772     7.205e-11, 6.616e-11, 6.036e-11, 5.475e-11, 5.007e-11, 4.638e-11,
00773     4.346e-11, 4.055e-11, 3.763e-11, 3.471e-11, 3.186e-11, 2.905e-11,
00774     2.631e-11, 2.358e-11, 2.415e-11, 2.949e-11, 3.952e-11, 5.155e-11,
00775     6.76e-11, 8.741e-11, 1.099e-10, 1.278e-10, 1.414e-10, 1.512e-10,
00776     1.607e-10, 1.699e-10, 1.774e-10, 1.832e-10, 1.871e-10, 1.907e-10,
00777     1.943e-10, 1.974e-10, 1.993e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00778     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00779     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00780     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00781     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00782     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00783     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10,
00784     2e-10, 2e-10, 2e-10, 2e-10, 2e-10, 2e-10
00785 };
00786
00787 static int ig_co2 = -999;
00788
00789 double *q[NG] = { NULL };
00790
00791 /* Find emitter index of CO2... */
00792 if (ig_co2 == -999)
00793     ig_co2 = find_emitter(ctl, "CO2");
00794
00795 /* Identify variable... */
00796 for (int ig = 0; ig < ctl->ng; ig++) {
00797     q[ig] = NULL;
00798     if (strcasecmp(ctl->emitter[ig], "C2H2") == 0)
00799         q[ig] = c2h2;
00800     if (strcasecmp(ctl->emitter[ig], "C2H6") == 0)
00801         q[ig] = c2h6;
00802     if (strcasecmp(ctl->emitter[ig], "CCl4") == 0)
00803         q[ig] = ccl4;
00804     if (strcasecmp(ctl->emitter[ig], "CH4") == 0)
00805         q[ig] = ch4;
00806     if (strcasecmp(ctl->emitter[ig], "ClO") == 0)
00807         q[ig] = clo;
00808     if (strcasecmp(ctl->emitter[ig], "ClONO2") == 0)
00809         q[ig] = clono2;
00810     if (strcasecmp(ctl->emitter[ig], "CO") == 0)
00811         q[ig] = co;
00812     if (strcasecmp(ctl->emitter[ig], "COF2") == 0)
00813         q[ig] = cof2;
00814     if (strcasecmp(ctl->emitter[ig], "F11") == 0)
00815         q[ig] = f11;
00816     if (strcasecmp(ctl->emitter[ig], "F12") == 0)
00817         q[ig] = f12;
00818     if (strcasecmp(ctl->emitter[ig], "F14") == 0)
00819         q[ig] = f14;
00820     if (strcasecmp(ctl->emitter[ig], "F22") == 0)
00821         q[ig] = f22;

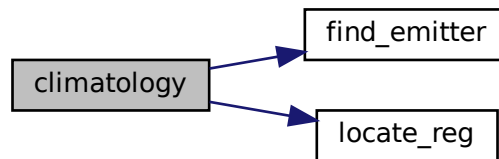
```

```

00822     if (strcasecmp(ctl->emitter[ig], "H2O") == 0)
00823         q[ig] = h2o;
00824     if (strcasecmp(ctl->emitter[ig], "H2O2") == 0)
00825         q[ig] = h2o2;
00826     if (strcasecmp(ctl->emitter[ig], "HCN") == 0)
00827         q[ig] = hcn;
00828     if (strcasecmp(ctl->emitter[ig], "HNO3") == 0)
00829         q[ig] = hno3;
00830     if (strcasecmp(ctl->emitter[ig], "HNO4") == 0)
00831         q[ig] = hno4;
00832     if (strcasecmp(ctl->emitter[ig], "HOCl") == 0)
00833         q[ig] = hocl;
00834     if (strcasecmp(ctl->emitter[ig], "N2O") == 0)
00835         q[ig] = n2o;
00836     if (strcasecmp(ctl->emitter[ig], "N2O5") == 0)
00837         q[ig] = n2o5;
00838     if (strcasecmp(ctl->emitter[ig], "NH3") == 0)
00839         q[ig] = nh3;
00840     if (strcasecmp(ctl->emitter[ig], "NO") == 0)
00841         q[ig] = no;
00842     if (strcasecmp(ctl->emitter[ig], "NO2") == 0)
00843         q[ig] = no2;
00844     if (strcasecmp(ctl->emitter[ig], "O3") == 0)
00845         q[ig] = o3;
00846     if (strcasecmp(ctl->emitter[ig], "OCS") == 0)
00847         q[ig] = ocs;
00848     if (strcasecmp(ctl->emitter[ig], "SF6") == 0)
00849         q[ig] = sf6;
00850     if (strcasecmp(ctl->emitter[ig], "SO2") == 0)
00851         q[ig] = so2;
00852 }
00853
00854 /* Loop over atmospheric data points... */
00855 for (int ip = 0; ip < atm->np; ip++) {
00856
00857     /* Get altitude index... */
00858     int iz = locate_reg(z, 121, atm->z[ip]);
00859
00860     /* Interpolate pressure... */
00861     atm->p[ip] = EXP(z[iz], pre[iz], z[iz + 1], pre[iz + 1], atm->z[ip]);
00862
00863     /* Interpolate temperature... */
00864     atm->t[ip] = LIN(z[iz], tem[iz], z[iz + 1], tem[iz + 1], atm->z[ip]);
00865
00866     /* Interpolate trace gases... */
00867     for (int ig = 0; ig < ctl->ng; ig++)
00868         if (q[ig] != NULL)
00869             atm->q[ig][ip] =
00870                 LIN(z[iz], q[ig][iz], z[iz + 1], q[ig][iz + 1], atm->z[ip]);
00871         else
00872             atm->q[ig][ip] = 0;
00873
00874     /* Set CO2... */
00875     if (ig_co2 >= 0)
00876         atm->q[ig_co2][ip] =
00877             371.789948e-6 + 2.026214e-6 * (atm->time[ip] - 63158400.) / 31557600.;
00878
00879     /* Set extinction to zero... */
00880     for (int iw = 0; iw < ctl->nw; iw++)
00881         atm->k[iw][ip] = 0;
00882
00883     /* Set cloud layer... */
00884     atm->clz = atm->cldz = 0;
00885     for (int icl = 0; icl < ctl->ncl; icl++)
00886         atm->clk[icl] = 0;
00887
00888     /* Set surface layer... */
00889     atm->sfz = atm->sfp = atm->sft = 0;
00890     for (int isf = 0; isf < ctl->nsf; isf++)
00891         atm->sfeps[isf] = 1;
00892 }
00893 }

```

Here is the call graph for this function:



```

5.19.3.6 ctmco2() double ctmco2 (
    double nu,
    double p,
    double t,
    double u )
  
```

Compute carbon dioxide continuum (optical depth).

Definition at line 897 of file [jurassic.c](#).

```

00901     {
00902
00903     static double co2296[2001] = { 9.3388e-5, 9.7711e-5, 1.0224e-4, 1.0697e-4,
00904     1.1193e-4, 1.1712e-4, 1.2255e-4, 1.2824e-4, 1.3419e-4, 1.4043e-4,
00905     1.4695e-4, 1.5378e-4, 1.6094e-4, 1.6842e-4, 1.7626e-4, 1.8447e-4,
00906     1.9307e-4, 2.0207e-4, 2.1149e-4, 2.2136e-4, 2.3169e-4, 2.4251e-4,
00907     2.5384e-4, 2.657e-4, 2.7813e-4, 2.9114e-4, 3.0477e-4, 3.1904e-4,
00908     3.3399e-4, 3.4965e-4, 3.6604e-4, 3.8322e-4, 4.0121e-4, 4.2006e-4,
00909     4.398e-4, 4.6047e-4, 4.8214e-4, 5.0483e-4, 5.286e-4, 5.535e-4,
00910     5.7959e-4, 6.0693e-4, 6.3557e-4, 6.6558e-4, 6.9702e-4, 7.2996e-4,
00911     7.6449e-4, 8.0066e-4, 8.3856e-4, 8.7829e-4, 9.1991e-4, 9.6354e-4,
00912     .0010093, .0010572, .0011074, .00116, .0012152, .001273,
00913     .0013336, .0013972, .0014638, .0015336, .0016068, .0016835,
00914     .001764, .0018483, .0019367, .0020295, .0021267, .0022286,
00915     .0023355, .0024476, .0025652, .0026885, .0028178, .0029534,
00916     .0030956, .0032448, .0034012, .0035654, .0037375, .0039181,
00917     .0041076, .0043063, .0045148, .0047336, .0049632, .005204,
00918     .0054567, .0057219, .0060002, .0062923, .0065988, .0069204,
00919     .007258, .0076123, .0079842, .0083746, .0087844, .0092146,
00920     .0096663, .01014, .010638, .011161, .01171, .012286, .012891,
00921     .013527, .014194, .014895, .015631, .016404, .017217, .01807,
00922     .018966, .019908, .020897, .021936, .023028, .024176, .025382,
00923     .026649, .027981, .02938, .030851, .032397, .034023, .035732,
00924     .037528, .039416, .041402, .04349, .045685, .047994, .050422,
00925     .052975, .055661, .058486, .061458, .064584, .067873, .071334,
00926     .074975, .078807, .082839, .087082, .091549, .096249, .1012,
00927     .10641, .11189, .11767, .12375, .13015, .13689, .14399, .15147,
00928     .15935, .16765, .17639, .18561, .19531, .20554, .21632, .22769,
00929     .23967, .25229, .2656, .27964, .29443, .31004, .3265, .34386,
00930     .36218, .3815, .40188, .42339, .44609, .47004, .49533, .52202,
00931     .5502, .57995, .61137, .64455, .6796, .71663, .75574, .79707,
00932     .84075, .88691, .9357, .98728, 1.0418, 1.0995, 1.1605, 1.225,
00933     1.2932, 1.3654, 1.4418, 1.5227, 1.6083, 1.6989, 1.7948, 1.8964,
00934     2.004, 2.118, 2.2388, 2.3668, 2.5025, 2.6463, 2.7988, 2.9606,
00935     3.1321, 3.314, 3.5071, 3.712, 3.9296, 4.1605, 4.4058, 4.6663,
00936     4.9431, 5.2374, 5.5501, 5.8818, 6.2353, 6.6114, 7.0115, 7.4372,
00937     7.8905, 8.3731, 8.8871, 9.4349, 10.019, 10.641, 11.305, 12.013,
00938     12.769, 13.576, 14.437, 15.358, 16.342, 17.39, 18.513, 19.716,
00939     21.003, 22.379, 23.854, 25.436, 27.126, 28.942, 30.89, 32.973,
00940     35.219, 37.634, 40.224, 43.021, 46.037, 49.29, 52.803, 56.447,
00941     60.418, 64.792, 69.526, 74.637, 80.182, 86.193, 92.713, 99.786,
00942     107.47, 115.84, 124.94, 134.86, 145.69, 157.49, 170.3, 184.39,
00943     199.83, 216.4, 234.55, 254.72, 276.82, 299.85, 326.16, 354.99,
00944     386.51, 416.68, 449.89, 490.12, 534.35, 578.25, 632.26, 692.61,
00945     756.43, 834.75, 924.11, 1016.9, 996.96, 1102.7, 1219.2, 1351.9,
  
```

```
00946 1494.3, 1654.1, 1826.5, 2027.9, 2249., 2453.8, 2714.4, 2999.4,
00947 3209.5, 3509., 3840.4, 3907.5, 4190.7, 4533.5, 4648.3, 5059.1,
00948 5561.6, 6191.4, 6820.8, 7905.9, 9362.2, 2431.3, 2211.3, 2046.8,
00949 2023.8, 1985.9, 1905.9, 1491.1, 1369.8, 1262.2, 1200.7, 887.74,
00950 820.25, 885.23, 887.21, 816.73, 1126.9, 1216.2, 1272.4, 1579.5,
00951 1634.2, 1656.3, 1657.9, 1789.5, 1670.8, 1509.5, 8474.6, 7489.2,
00952 6793.6, 6117., 5574.1, 5141.2, 5084.6, 4745.1, 4413.2, 4102.8,
00953 4024.7, 3715., 3398.6, 3100.8, 2900.4, 2629.2, 2374., 2144.7,
00954 1955.8, 1760.8, 1591.2, 1435.2, 1296.2, 1174., 1065.1, 967.76,
00955 999.48, 897.45, 809.23, 732.77, 670.26, 611.93, 560.11, 518.77,
00956 476.84, 438.8, 408.48, 380.21, 349.24, 322.71, 296.65, 272.85,
00957 251.96, 232.04, 213.88, 197.69, 182.41, 168.41, 155.79, 144.05,
00958 133.31, 123.48, 114.5, 106.21, 98.591, 91.612, 85.156, 79.204,
00959 73.719, 68.666, 63.975, 59.637, 56.35, 52.545, 49.042, 45.788,
00960 42.78, 39.992, 37.441, 35.037, 32.8, 30.744, 28.801, 26.986,
00961 25.297, 23.731, 22.258, 20.883, 19.603, 18.403, 17.295, 16.249,
00962 15.271, 14.356, 13.501, 12.701, 11.954, 11.254, 10.6, 9.9864,
00963 9.4118, 8.8745, 8.3714, 7.8997, 7.4578, 7.0446, 6.6573, 6.2949,
00964 5.9577, 5.6395, 5.3419, 5.063, 4.8037, 4.5608, 4.3452, 4.1364,
00965 3.9413, 3.7394, 3.562, 3.3932, 3.2325, 3.0789, 2.9318, 2.7898,
00966 2.6537, 2.5225, 2.3958, 2.2305, 2.1215, 2.0245, 1.9427, 1.8795,
00967 1.8336, 1.7604, 1.7016, 1.6419, 1.5282, 1.4611, 1.3443, 1.27,
00968 1.1675, 1.0824, 1.0534, .99833, .95854, .92981, .90887, .89346,
00969 .88113, .87068, .86102, .85096, .88262, .86151, .83565, .80518,
00970 .77045, .73736, .74744, .74954, .75773, .82267, .83493, .89402,
00971 .89725, .93426, .95564, .94045, .94174, .93404, .92035, .90456,
00972 .88621, .86673, .78117, .7515, .72056, .68822, .65658, .62764,
00973 .55984, .55598, .57407, .60963, .63763, .66198, .61132, .60972,
00974 .52496, .50649, .41872, .3964, .32422, .27276, .24048, .23772,
00975 .2286, .22711, .23999, .32038, .34371, .36621, .38561, .39953,
00976 .40636, .44913, .42716, .3919, .35477, .33935, .3351, .39746,
00977 .40993, .49398, .49956, .56157, .54742, .57295, .57386, .55417,
00978 .50745, .471, .43446, .39102, .34993, .31269, .27888, .24912,
00979 .22291, .19994, .17972, .16197, .14633, .13252, .12029, .10942,
00980 .099745, .091118, .083404, .076494, .070292, .064716, .059697,
00981 .055173, .051093, .047411, .044089, .041092, .038392, .035965,
00982 .033789, .031846, .030122, .028607, .02729, .026169, .025209,
00983 .024405, .023766, .023288, .022925, .022716, .022681, .022685,
00984 .022768, .023133, .023325, .023486, .024004, .024126, .024083,
00985 .023785, .024023, .023029, .021649, .021108, .019454, .017809,
00986 .017292, .016635, .017037, .018068, .018977, .018756, .017847,
00987 .016557, .016142, .014459, .012869, .012381, .010875, .0098701,
00988 .009285, .0091698, .0091701, .0096145, .010553, .01106, .012613,
00989 .014362, .015017, .016507, .017741, .01768, .017784, .0171,
00990 .016357, .016172, .017257, .018978, .020935, .021741, .023567,
00991 .025183, .025589, .026732, .027648, .028278, .028215, .02856,
00992 .029015, .029062, .028851, .028497, .027825, .027801, .026523,
00993 .02487, .022967, .022168, .020194, .018605, .017903, .018439,
00994 .019697, .020311, .020855, .020057, .018608, .016738, .015963,
00995 .013844, .011801, .011134, .0097573, .0086007, .0086226,
00996 .0083721, .0090978, .0097616, .0098426, .011317, .012853, .01447,
00997 .014657, .015771, .016351, .016079, .014829, .013431, .013185,
00998 .013207, .01448, .016176, .017971, .018265, .019526, .020455,
00999 .019797, .019802, .0194, .018176, .017505, .016197, .015339,
01000 .014401, .013213, .012203, .011186, .010236, .0093288, .0084854,
01001 .0076837, .0069375, .0062614, .0056628, .0051153, .0046015,
01002 .0041501, .003752, .0033996, .0030865, .0028077, .0025586,
01003 .0023355, .0021353, .0019553, .0017931, .0016466, .0015141,
01004 .0013941, .0012852, .0011862, .0010962, .0010142, 9.3935e-4,
01005 8.71e-4, 8.0851e-4, 7.5132e-4, 6.9894e-4, 6.5093e-4, 6.0689e-4,
01006 5.6647e-4, 5.2935e-4, 4.9525e-4, 4.6391e-4, 4.3509e-4, 4.086e-4,
01007 3.8424e-4, 3.6185e-4, 3.4126e-4, 3.2235e-4, 3.0498e-4, 2.8904e-4,
01008 2.7444e-4, 2.6106e-4, 2.4883e-4, 2.3766e-4, 2.275e-4, 2.1827e-4,
01009 2.0992e-4, 2.0239e-4, 1.9563e-4, 1.896e-4, 1.8427e-4, 1.796e-4,
01010 1.7555e-4, 1.7209e-4, 1.692e-4, 1.6687e-4, 1.6505e-4, 1.6375e-4,
01011 1.6294e-4, 1.6261e-4, 1.6274e-4, 1.6334e-4, 1.6438e-4, 1.6587e-4,
01012 1.678e-4, 1.7017e-4, 1.7297e-4, 1.762e-4, 1.7988e-4, 1.8399e-4,
01013 1.8855e-4, 1.9355e-4, 1.9902e-4, 2.0494e-4, 2.1134e-4, 2.1823e-4,
01014 2.2561e-4, 2.335e-4, 2.4192e-4, 2.5088e-4, 2.604e-4, 2.705e-4,
01015 2.8119e-4, 2.9251e-4, 3.0447e-4, 3.171e-4, 3.3042e-4, 3.4447e-4,
01016 3.5927e-4, 3.7486e-4, 3.9127e-4, 4.0854e-4, 4.267e-4, 4.4579e-4,
01017 4.6586e-4, 4.8696e-4, 5.0912e-4, 5.324e-4, 5.5685e-4, 5.8253e-4,
01018 6.0949e-4, 6.378e-4, 6.6753e-4, 6.9873e-4, 7.3149e-4, 7.6588e-4,
01019 8.0198e-4, 8.3987e-4, 8.7964e-4, 9.2139e-4, 9.6522e-4, .0010112,
01020 .0010595, .0011102, .0011634, .0012193, .001278, .0013396,
01021 .0014043, .0014722, .0015436, .0016185, .0016972, .0017799,
01022 .0018668, .001958, .0020539, .0021547, .0022606, .0023719,
01023 .002489, .002612, .0027414, .0028775, .0030206, .0031712,
01024 .0033295, .0034962, .0036716, .0038563, .0040506, .0042553,
01025 .0044709, .004698, .0049373, .0051894, .0054552, .0057354,
01026 .006031, .0063427, .0066717, .0070188, .0073854, .0077726,
01027 .0081816, .0086138, .0090709, .0095543, .010066, .010607,
01028 .011181, .011789, .012433, .013116, .013842, .014613, .015432,
01029 .016304, .017233, .018224, .019281, .020394, .021574, .022836,
01030 .024181, .025594, .027088, .028707, .030401, .032245, .034219,
01031 .036262, .038539, .040987, .043578, .04641, .04949, .052726,
01032 .056326, .0602, .064093, .068521, .073278, .077734, .083064,
```



01033 .088731, .093885, .1003, .1072, .11365, .12187, .13078, .13989,  
01034 .15095, .16299, .17634, .19116, .20628, .22419, .24386, .26587,  
01035 .28811, .31399, .34321, .36606, .39675, .42742, .44243, .47197,  
01036 .49993, .49027, .51147, .52803, .48931, .49729, .5026, .43854,  
01037 .441, .44766, .43414, .46151, .50029, .55247, .43855, .32115,  
01038 .32607, .3431, .36119, .38029, .41179, .43996, .47144, .51853,  
01039 .55362, .59122, .66338, .69877, .74001, .82923, .86907, .90361,  
01040 1.0025, 1.031, 1.0559, 1.104, 1.1178, 1.1341, 1.1547, 1.351,  
01041 1.4772, 1.4812, 1.4907, 1.512, 1.5442, 1.5853, 1.6358, 1.6963,  
01042 1.7674, 1.8474, 1.9353, 2.0335, 2.143, 2.2592, 2.3853, 2.5217,  
01043 2.6686, 2.8273, 2.9998, 3.183, 3.3868, 3.6109, 3.8564, 4.1159,  
01044 4.4079, 4.7278, 5.0497, 5.3695, 5.758, 6.0834, 6.4976, 6.9312,  
01045 7.38, 7.5746, 7.9833, 8.3791, 8.3956, 8.7501, 9.1067, 9.072,  
01046 9.4649, 9.9112, 10.402, 10.829, 11.605, 12.54, 12.713, 10.443,  
01047 10.825, 11.375, 11.955, 12.623, 13.326, 14.101, 15.041, 15.547,  
01048 16.461, 17.439, 18.716, 19.84, 21.036, 22.642, 23.901, 25.244,  
01049 27.03, 28.411, 29.871, 31.403, 33.147, 34.744, 36.456, 39.239,  
01050 43.605, 45.162, 47.004, 49.093, 51.391, 53.946, 56.673, 59.629,  
01051 63.167, 66.576, 70.254, 74.222, 78.477, 83.034, 87.914, 93.18,  
01052 98.77, 104.74, 111.15, 117.95, 125.23, 133.01, 141.33, 150.21,  
01053 159.71, 169.89, 180.93, 192.54, 204.99, 218.34, 232.65, 248.,  
01054 264.47, 282.14, 301.13, 321.53, 343.48, 367.08, 392.5, 419.88,  
01055 449.4, 481.26, 515.64, 552.79, 592.99, 636.48, 683.61, 734.65,  
01056 789.99, 850.02, 915.14, 985.81, 1062.5, 1147.1, 1237.8, 1336.4,  
01057 1443.2, 1558.9, 1684.2, 1819.2, 1965.2, 2122.6, 2291.7, 2470.8,  
01058 2665.7, 2874.9, 3099.4, 3337.9, 3541., 3813.3, 4111.9, 4439.3,  
01059 4798.9, 5196., 5639.2, 6087.5, 6657.7, 7306.7, 8040.7, 8845.5,  
01060 9702.2, 10670., 11739., 12842., 14141., 15498., 17068., 18729.,  
01061 20557., 22559., 25248., 27664., 30207., 32915., 35611., 38081.,  
01062 40715., 43191., 41651., 42750., 43785., 44353., 44366., 44189.,  
01063 43618., 42862., 41878., 35133., 35215., 36383., 39420., 44055.,  
01064 44155., 45850., 46853., 39197., 38274., 29942., 28553., 21792.,  
01065 21228., 17106., 14955., 18181., 19557., 21427., 23728., 26301.,  
01066 28584., 30775., 32536., 33867., 40089., 39204., 37329., 34452.,  
01067 31373., 33921., 34800., 36043., 44415., 45162., 52181., 50895.,  
01068 54140., 50840., 50468., 48302., 44915., 40910., 36754., 32755.,  
01069 29093., 25860., 22962., 20448., 18247., 16326., 14645., 13165.,  
01070 11861., 10708., 9686.9, 8779.7, 7971.9, 7250.8, 6605.7, 6027.2,  
01071 5507.3, 5039.1, 4616.6, 4234.8, 3889., 3575.4, 3290.5, 3031.3,  
01072 2795.2, 2579.9, 2383.1, 2203.3, 2038.6, 1887.6, 1749.1, 1621.9,  
01073 1505., 1397.4, 1298.3, 1207., 1122.8, 1045., 973.1, 906.64,  
01074 845.16, 788.22, 735.48, 686.57, 641.21, 599.1, 559.99, 523.64,  
01075 489.85, 458.42, 429.16, 401.92, 376.54, 352.88, 330.82, 310.24,  
01076 291.03, 273.09, 256.34, 240.69, 226.05, 212.37, 199.57, 187.59,  
01077 176.37, 165.87, 156.03, 146.82, 138.17, 130.07, 122.47, 115.34,  
01078 108.65, 102.37, 96.473, 90.934, 85.73, 80.84, 76.243, 71.922,  
01079 67.858, 64.034, 60.438, 57.052, 53.866, 50.866, 48.04, 45.379,  
01080 42.872, 40.51, 38.285, 36.188, 34.211, 32.347, 30.588, 28.929,  
01081 27.362, 25.884, 24.489, 23.171, 21.929, 20.755, 19.646, 18.599,  
01082 17.61, 16.677, 15.795, 14.961, 14.174, 13.43, 12.725, 12.06,  
01083 11.431, 10.834, 10.27, 9.7361, 9.2302, 8.7518, 8.2997, 7.8724,  
01084 7.4674, 7.0848, 6.7226, 6.3794, 6.054, 5.745, 5.4525, 5.1752,  
01085 4.9121, 4.6625, 4.4259, 4.2015, 3.9888, 3.7872, 3.5961, 3.4149,  
01086 3.2431, 3.0802, 2.9257, 2.7792, 2.6402, 2.5084, 2.3834, 2.2648,  
01087 2.1522, 2.0455, 1.9441, 1.848, 1.7567, 1.6701, 1.5878, 1.5097,  
01088 1.4356, 1.3651, 1.2981, 1.2345, 1.174, 1.1167, 1.062, 1.0101,  
01089 .96087, .91414, .86986, .82781, .78777, .74971, .71339, .67882,  
01090 .64604, .61473, .58507, .55676, .52987, .5044, .48014, .45715,  
01091 .43527, .41453, .3948, .37609, .35831, .34142, .32524, .30995,  
01092 .29536, .28142, .26807, .25527, .24311, .23166, .22077, .21053,  
01093 .20081, .19143, .18261, .17407, .16603, .15833, .15089, .14385,  
01094 .13707, .13065, .12449, .11865, .11306, .10774, .10266, .097818,  
01095 .093203, .088815, .084641, .080671, .076892, .073296, .069873,  
01096 .066613, .06351, .060555, .05774, .055058, .052504, .050071,  
01097 .047752, .045543, .043438, .041432, .039521, .037699, .035962,  
01098 .034307, .032729, .031225, .029791, .028423, .02712, .025877,  
01099 .024692, .023563, .022485, .021458, .020478, .019543, .018652,  
01100 .017806, .016992, .016219, .015481, .014778, .014107, .013467,  
01101 .012856, .012274, .011718, .011188, .010682, .0102, .0097393,  
01102 .0093001, .008881, .0084812, .0080997, .0077358, .0073885,  
01103 .0070571, .0067409, .0064393, .0061514, .0058768, .0056147,  
01104 .0053647, .0051262, .0048987, .0046816, .0044745, .0042769,  
01105 .0040884, .0039088, .0037373, .0035739, .003418, .0032693,  
01106 .0031277, .0029926, .0028639, .0027413, .0026245, .0025133,  
01107 .0024074, .0023066, .0022108, .0021196, .002033, .0019507,  
01108 .0018726, .0017985, .0017282, .0016617, .0015988, .0015394,  
01109 .0014834, .0014306, .0013811, .0013346, .0012911, .0012506,  
01110 .0012131, .0011784, .0011465, .0011175, .0010912, .0010678,  
01111 .0010472, .0010295, .0010147, .001003, 9.9428e-4, 9.8883e-4,  
01112 9.8673e-4, 9.8821e-4, 9.9343e-4, .0010027, .0010164, .0010348,  
01113 .0010586, .0010882, .0011245, .0011685, .0012145, .0012666,  
01114 .0013095, .0013688, .0014048, .0014663, .0015309, .0015499,  
01115 .0016144, .0016312, .001705, .0017892, .0018499, .0019715,  
01116 .0021102, .0022442, .0024284, .0025893, .0027703, .0029445,  
01117 .0031193, .003346, .0034552, .0036906, .0037584, .0040084,  
01118 .0041934, .0044587, .0047093, .0049759, .0053421, .0055134,  
01119 .0059048, .0058663, .0061036, .0063259, .0059657, .0060653,

```
01120 .0060972, .0055539, .0055653, .0055772, .005331, .0054953,
01121 .0055919, .0058684, .006183, .0066675, .0069808, .0075142,
01122 .0078536, .0084282, .0089454, .0094625, .0093703, .0095857,
01123 .0099283, .010063, .010521, .0097778, .0098175, .010379, .010447,
01124 .0105, .010617, .010706, .01078, .011177, .011212, .011304,
01125 .011446, .011603, .011816, .012165, .012545, .013069, .013539,
01126 .01411, .014776, .016103, .017016, .017994, .018978, .01998,
01127 .021799, .022745, .023681, .024627, .025562, .026992, .027958,
01128 .029013, .030154, .031402, .03228, .033651, .035272, .037088,
01129 .039021, .041213, .043597, .045977, .04877, .051809, .054943,
01130 .058064, .061528, .06537, .069309, .071928, .075752, .079589,
01131 .083352, .084096, .087497, .090817, .091198, .094966, .099045,
01132 .10429, .10867, .11518, .12269, .13126, .14087, .15161, .16388,
01133 .16423, .1759, .18721, .19994, .21275, .22513, .23041, .24231,
01134 .25299, .25396, .26396, .27696, .27929, .2908, .30595, .31433,
01135 .3282, .3429, .35944, .37467, .39277, .41245, .43326, .45649,
01136 .48152, .51897, .54686, .57877, .61263, .64962, .68983, .73945,
01137 .78619, .83537, .89622, .95002, 1.0067, 1.0742, 1.1355, 1.2007,
01138 1.2738, 1.347, 1.4254, 1.5094, 1.6009, 1.6976, 1.8019, 1.9148,
01139 2.0357, 2.166, 2.3066, 2.4579, 2.6208, 2.7966, 2.986, 3.188,
01140 3.4081, 3.6456, 3.9, 4.1747, 4.4712, 4.7931, 5.1359, 5.5097,
01141 5.9117, 6.3435, 6.8003, 7.3001, 7.8385, 8.3945, 9.011, 9.6869,
01142 10.392, 11.18, 12.036, 12.938, 13.944, 14.881, 16.029, 17.255,
01143 18.574, 19.945, 21.38, 22.9, 24.477, 26.128, 27.87, 29.037,
01144 30.988, 33.145, 35.506, 37.76, 40.885, 44.487, 48.505, 52.911,
01145 57.56, 61.964, 67.217, 72.26, 78.343, 85.08, 91.867, 99.435,
01146 107.68, 116.97, 127.12, 138.32, 150.26, 163.04, 174.81, 189.26,
01147 205.61, 224.68, 240.98, 261.88, 285.1, 307.58, 334.35, 363.53,
01148 394.68, 427.85, 458.85, 489.25, 472.87, 486.93, 496.27, 501.52,
01149 501.57, 497.14, 488.09, 476.32, 393.76, 388.51, 393.42, 414.45,
01150 455.12, 514.62, 520.38, 547.42, 562.6, 487.47, 480.83, 391.06,
01151 376.92, 303.7, 295.91, 256.03, 236.73, 280.38, 310.71, 335.53,
01152 367.88, 401.94, 435.52, 469.13, 497.94, 588.82, 597.94, 597.2,
01153 588.28, 571.2, 555.75, 603.56, 638.15, 680.75, 801.72, 848.01,
01154 962.15, 990.06, 1068.1, 1076.2, 1115.3, 1134.2, 1136.6, 1119.1,
01155 1108.9, 1090.6, 1068.7, 1041.9, 1005.4, 967.98, 927.08, 780.1,
01156 751.41, 733.12, 742.65, 785.56, 855.16, 852.45, 878.1, 784.59,
01157 777.81, 765.13, 622.93, 498.09, 474.89, 386.9, 378.48, 336.17,
01158 322.04, 329.57, 350.5, 383.38, 420.02, 462.39, 499.71, 531.98,
01159 654.99, 653.43, 639.99, 605.16, 554.16, 504.42, 540.64, 552.33,
01160 679.46, 699.51, 713.91, 832.17, 919.91, 884.96, 907.57, 846.56,
01161 818.56, 768.93, 706.71, 642.17, 575.95, 515.38, 459.07, 409.02,
01162 364.61, 325.46, 291.1, 260.89, 234.39, 211.01, 190.38, 172.11,
01163 155.91, 141.49, 128.63, 117.13, 106.84, 97.584, 89.262, 81.756,
01164 74.975, 68.842, 63.28, 58.232, 53.641, 49.46, 45.649, 42.168,
01165 38.991, 36.078, 33.409, 30.96, 28.71, 26.642, 24.737, 22.985,
01166 21.37, 19.882, 18.512, 17.242, 16.073, 14.987, 13.984, 13.05,
01167 12.186, 11.384, 10.637, 9.9436, 9.2988, 8.6991, 8.141, 7.6215,
01168 7.1378, 6.6872, 6.2671, 5.8754, 5.51, 5.1691, 4.851, 4.5539,
01169 4.2764, 4.0169, 3.7742, 3.5472, 3.3348, 3.1359, 2.9495, 2.7749,
01170 2.6113, 2.4578, 2.3139, 2.1789, 2.0523, 1.9334, 1.8219, 1.7171,
01171 1.6188, 1.5263, 1.4395, 1.3579, 1.2812, 1.209, 1.1411, 1.0773,
01172 1.0171, .96048, .90713, .85684, .80959, .76495, .72282, .68309,
01173 .64563, .61035, .57707, .54573, .51622, .48834, .46199, .43709,
01174 .41359, .39129, .37034, .35064, .33198, .31442, .29784, .28218,
01175 .26732, .25337, .24017, .22774, .21601, .20479, .19426
01176 };
01177
01178 static double co2260[2001] = { 5.7971e-5, 6.0733e-5, 6.3628e-5, 6.6662e-5,
01179 6.9843e-5, 7.3176e-5, 7.6671e-5, 8.0334e-5, 8.4175e-5, 8.8201e-5,
01180 9.2421e-5, 9.6846e-5, 1.0149e-4, 1.0635e-4, 1.1145e-4, 1.1679e-4,
01181 1.224e-4, 1.2828e-4, 1.3444e-4, 1.409e-4, 1.4768e-4, 1.5479e-4,
01182 1.6224e-4, 1.7006e-4, 1.7826e-4, 1.8685e-4, 1.9587e-4, 2.0532e-4,
01183 2.1524e-4, 2.2565e-4, 2.3656e-4, 2.48e-4, 2.6001e-4, 2.7261e-4,
01184 2.8582e-4, 2.9968e-4, 3.1422e-4, 3.2948e-4, 3.4548e-4, 3.6228e-4,
01185 3.799e-4, 3.9838e-4, 4.1778e-4, 4.3814e-4, 4.595e-4, 4.8191e-4,
01186 5.0543e-4, 5.3012e-4, 5.5603e-4, 5.8321e-4, 6.1175e-4, 6.417e-4,
01187 6.7314e-4, 7.0614e-4, 7.4078e-4, 7.7714e-4, 8.1531e-4, 8.5538e-4,
01188 8.9745e-4, 9.4162e-4, 9.8798e-4, .0010367, .0010878, .0011415,
01189 .0011978, .001257, .0013191, .0013844, .001453, .0015249,
01190 .0016006, .00168, .0017634, .001851, .001943, .0020397, .0021412,
01191 .0022479, .00236, .0024778, .0026015, .0027316, .0028682,
01192 .0030117, .0031626, .0033211, .0034877, .0036628, .0038469,
01193 .0040403, .0042436, .0044574, .004682, .0049182, .0051665,
01194 .0054276, .0057021, .0059907, .0062942, .0066133, .0069489,
01195 .0073018, .0076729, .0080632, .0084738, .0089056, .0093599,
01196 .0098377, .01034, .010869, .011426, .012011, .012627, .013276,
01197 .013958, .014676, .015431, .016226, .017063, .017944, .018872,
01198 .019848, .020876, .021958, .023098, .024298, .025561, .026892,
01199 .028293, .029769, .031323, .032961, .034686, .036503, .038418,
01200 .040435, .042561, .044801, .047161, .049649, .052271, .055035,
01201 .057948, .061019, .064256, .06767, .07127, .075066, .079069,
01202 .083291, .087744, .092441, .097396, .10262, .10814, .11396,
01203 .1201, .12658, .13342, .14064, .14826, .1563, .1648, .17376,
01204 .18323, .19324, .2038, .21496, .22674, .23919, .25234, .26624,
01205 .28093, .29646, .31287, .33021, .34855, .36794, .38844, .41012,
01206 .43305, .45731, .48297, .51011, .53884, .56924, .60141, .63547,
```

01207 .67152, .70969, .75012, .79292, .83826, .8863, .93718, .99111,  
01208 1.0482, 1.1088, 1.173, 1.2411, 1.3133, 1.3898, 1.471, 1.5571,  
01209 1.6485, 1.7455, 1.8485, 1.9577, 2.0737, 2.197, 2.3278, 2.4668,  
01210 2.6145, 2.7715, 2.9383, 3.1156, 3.3042, 3.5047, 3.7181, 3.9451,  
01211 4.1866, 4.4437, 4.7174, 5.0089, 5.3192, 5.65, 6.0025, 6.3782,  
01212 6.7787, 7.206, 7.6617, 8.1479, 8.6669, 9.221, 9.8128, 10.445,  
01213 11.12, 11.843, 12.615, 13.441, 14.325, 15.271, 16.283, 17.367,  
01214 18.529, 19.776, 21.111, 22.544, 24.082, 25.731, 27.504, 29.409,  
01215 31.452, 33.654, 36.024, 38.573, 41.323, 44.29, 47.492, 50.951,  
01216 54.608, 58.588, 62.929, 67.629, 72.712, 78.226, 84.207, 90.699,  
01217 97.749, 105.42, 113.77, 122.86, 132.78, 143.61, 155.44, 168.33,  
01218 182.48, 198.01, 214.87, 233.39, 253.86, 276.34, 300.3, 327.28,  
01219 356.89, 389.48, 422.29, 458.99, 501.39, 548.13, 595.62, 652.74,  
01220 716.54, 784.57, 866.78, 960.59, 1062.8, 1072.5, 1189.5, 1319.4,  
01221 1467.6, 1630.2, 1813.7, 2016.9, 2253., 2515.3, 2773.5, 3092.8,  
01222 3444.4, 3720.4, 4104.3, 4527.5, 4645.9, 5021.7, 5462.2, 5597.,  
01223 6110.6, 6732.5, 7513.8, 8270.6, 9640.6, 11487., 2796.1, 2680.1,  
01224 2441.6, 2404.2, 2334.8, 2215.2, 1642.5, 1477.9, 1328.1, 1223.5,  
01225 843.34, 766.96, 831.65, 834.84, 774.85, 1156.3, 1275.6, 1366.1,  
01226 1795.6, 1885., 1936.5, 1953.4, 2154.4, 2002.7, 1789.8, 10381.,  
01227 9040., 8216.5, 7384.7, 6721.9, 6187.7, 6143.8, 5703.9, 5276.6,  
01228 4873.1, 4736., 4325.3, 3927., 3554.1, 3286.1, 2950.1, 2642.4,  
01229 2368.7, 2138.9, 1914., 1719.6, 1543.9, 1388.6, 1252.1, 1132.2,  
01230 1024.1, 1025.4, 920.58, 829.59, 750.54, 685.01, 624.25, 570.14,  
01231 525.81, 481.85, 441.95, 408.71, 377.23, 345.86, 318.51, 292.26,  
01232 268.34, 247.04, 227.14, 209.02, 192.69, 177.59, 163.78, 151.26,  
01233 139.73, 129.19, 119.53, 110.7, 102.57, 95.109, 88.264, 81.948,  
01234 76.13, 70.768, 65.827, 61.251, 57.022, 53.495, 49.824, 46.443,  
01235 43.307, 40.405, 37.716, 35.241, 32.923, 30.77, 28.78, 26.915,  
01236 25.177, 23.56, 22.059, 20.654, 19.345, 18.126, 16.988, 15.93,  
01237 14.939, 14.014, 13.149, 12.343, 11.589, 10.884, 10.225, 9.6093,  
01238 9.0327, 8.4934, 7.9889, 7.5166, 7.0744, 6.6604, 6.2727, 5.9098,  
01239 5.5701, 5.2529, 4.955, 4.676, 4.4148, 4.171, 3.9426, 3.7332,  
01240 3.5347, 3.3493, 3.1677, 3.0025, 2.8466, 2.6994, 2.5601, 2.4277,  
01241 2.3016, 2.1814, 2.0664, 1.9564, 1.8279, 1.7311, 1.6427, 1.5645,  
01242 1.4982, 1.443, 1.374, 1.3146, 1.2562, 1.17, 1.1105, 1.0272,  
01243 .96863, .89718, .83654, .80226, .75908, .72431, .69573, .67174,  
01244 .65126, .63315, .61693, .60182, .58715, .59554, .57649, .55526,  
01245 .53177, .50622, .48176, .4813, .47642, .47492, .50273, .50293,  
01246 .52687, .52239, .53419, .53814, .52626, .52211, .51492, .50622,  
01247 .49746, .48841, .4792, .43534, .41999, .40349, .38586, .36799,  
01248 .35108, .31089, .30803, .3171, .33599, .35041, .36149, .32924,  
01249 .32462, .27309, .25961, .20922, .19504, .15683, .13098, .11588,  
01250 .11478, .11204, .11363, .12135, .16423, .17785, .19094, .20236,  
01251 .21084, .2154, .24108, .22848, .20871, .18797, .17963, .17834,  
01252 .21552, .22284, .26945, .27052, .30108, .28977, .29772, .29224,  
01253 .27658, .24956, .22777, .20654, .18392, .16338, .1452, .12916,  
01254 .1152, .10304, .092437, .083163, .075031, .067878, .061564,  
01255 .055976, .051018, .046609, .042679, .03917, .036032, .033223,  
01256 .030706, .02845, .026428, .024617, .022998, .021554, .02027,  
01257 .019136, .018141, .017278, .016541, .015926, .015432, .015058,  
01258 .014807, .014666, .014635, .014728, .014947, .01527, .015728,  
01259 .016345, .017026, .017798, .018839, .019752, .020636, .021886,  
01260 .022695, .02327, .023478, .024292, .023544, .022222, .021932,  
01261 .020052, .018143, .017722, .017031, .017782, .01938, .020734,  
01262 .020476, .019255, .017477, .016878, .014617, .012489, .011765,  
01263 .0099077, .0086446, .0079446, .0078644, .0079763, .008671,  
01264 .01001, .0108, .012933, .015349, .016341, .018484, .020254,  
01265 .020254, .020478, .019591, .018595, .018385, .019913, .022254,  
01266 .024847, .025809, .028053, .029924, .030212, .031367, .03222,  
01267 .032739, .032537, .03286, .033344, .033507, .033499, .033339,  
01268 .032809, .033041, .031723, .029837, .027511, .026603, .024032,  
01269 .021914, .020948, .021701, .023425, .024259, .024987, .023818,  
01270 .021768, .019223, .018144, .015282, .012604, .01163, .0097907,  
01271 .008336, .0082473, .0079582, .0088077, .009779, .010129, .012145,  
01272 .014378, .016761, .01726, .018997, .019998, .019809, .01819,  
01273 .016358, .016099, .01617, .017939, .020223, .022521, .02277,  
01274 .024279, .025247, .024222, .023989, .023224, .021493, .020362,  
01275 .018596, .017309, .015975, .014466, .013171, .011921, .01078,  
01276 .0097229, .0087612, .0078729, .0070682, .0063494, .0057156,  
01277 .0051459, .0046273, .0041712, .0037686, .0034119, .003095,  
01278 .0028126, .0025603, .0023342, .0021314, .0019489, .0017845,  
01279 .001636, .0015017, .00138, .0012697, .0011694, .0010782,  
01280 9.9507e-4, 9.1931e-4, 8.5013e-4, 7.869e-4, 7.2907e-4, 6.7611e-4,  
01281 6.2758e-4, 5.8308e-4, 5.4223e-4, 5.0473e-4, 4.7027e-4, 4.3859e-4,  
01282 4.0946e-4, 3.8265e-4, 3.5798e-4, 3.3526e-4, 3.1436e-4, 2.9511e-4,  
01283 2.7739e-4, 2.6109e-4, 2.4609e-4, 2.3229e-4, 2.1961e-4, 2.0797e-4,  
01284 1.9729e-4, 1.875e-4, 1.7855e-4, 1.7038e-4, 1.6294e-4, 1.5619e-4,  
01285 1.5007e-4, 1.4456e-4, 1.3961e-4, 1.3521e-4, 1.3131e-4, 1.2789e-4,  
01286 1.2494e-4, 1.2242e-4, 1.2032e-4, 1.1863e-4, 1.1733e-4, 1.1641e-4,  
01287 1.1585e-4, 1.1565e-4, 1.158e-4, 1.1629e-4, 1.1712e-4, 1.1827e-4,  
01288 1.1976e-4, 1.2158e-4, 1.2373e-4, 1.262e-4, 1.2901e-4, 1.3214e-4,  
01289 1.3562e-4, 1.3944e-4, 1.4361e-4, 1.4814e-4, 1.5303e-4, 1.5829e-4,  
01290 1.6394e-4, 1.6999e-4, 1.7644e-4, 1.8332e-4, 1.9063e-4, 1.984e-4,  
01291 2.0663e-4, 2.1536e-4, 2.246e-4, 2.3436e-4, 2.4468e-4, 2.5558e-4,  
01292 2.6708e-4, 2.7921e-4, 2.92e-4, 3.0548e-4, 3.1968e-4, 3.3464e-4,  
01293 3.5039e-4, 3.6698e-4, 3.8443e-4, 4.0281e-4, 4.2214e-4, 4.4248e-4,

```

01294 4.6389e-4, 4.864e-4, 5.1009e-4, 5.3501e-4, 5.6123e-4, 5.888e-4,
01295 6.1781e-4, 6.4833e-4, 6.8043e-4, 7.142e-4, 7.4973e-4, 7.8711e-4,
01296 8.2644e-4, 8.6783e-4, 9.1137e-4, 9.5721e-4, .0010054, .0010562,
01297 .0011096, .0011659, .0012251, .0012875, .0013532, .0014224,
01298 .0014953, .001572, .0016529, .0017381, .0018279, .0019226,
01299 .0020224, .0021277, .0022386, .0023557, .0024792, .0026095,
01300 .002747, .0028921, .0030453, .0032071, .003378, .0035586,
01301 .0037494, .003951, .0041642, .0043897, .0046282, .0048805,
01302 .0051476, .0054304, .00573, .0060473, .0063837, .0067404,
01303 .0071188, .0075203, .0079466, .0083994, .0088806, .0093922,
01304 .0099366, .010516, .011134, .011792, .012494, .013244, .014046,
01305 .014898, .015808, .016781, .017822, .018929, .020108, .02138,
01306 .022729, .02419, .02576, .027412, .029233, .031198, .033301,
01307 .035594, .038092, .040767, .04372, .046918, .050246, .053974,
01308 .058009, .061976, .066586, .071537, .076209, .081856, .087998,
01309 .093821, .10113, .10913, .11731, .12724, .13821, .15025, .1639,
01310 .17807, .19472, .21356, .23496, .25758, .28387, .31389, .34104,
01311 .37469, .40989, .43309, .46845, .5042, .5023, .52981, .55275,
01312 .51075, .51976, .52457, .44779, .44721, .4503, .4243, .45244,
01313 .49491, .55399, .39021, .24802, .2501, .2618, .27475, .28879,
01314 .31317, .33643, .36257, .4018, .43275, .46525, .53333, .56599,
01315 .60557, .70142, .74194, .77736, .88567, .91182, .93294, .98407,
01316 .98772, .99176, .9995, 1.2405, 1.3602, 1.338, 1.3255, 1.3267,
01317 1.3404, 1.3634, 1.3967, 1.4407, 1.4961, 1.5603, 1.6328, 1.7153,
01318 1.8094, 1.9091, 2.018, 2.1367, 2.264, 2.4035, 2.5562, 2.7179,
01319 2.9017, 3.1052, 3.3304, 3.5731, 3.8488, 4.1553, 4.4769, 4.7818,
01320 5.1711, 5.5204, 5.9516, 6.4097, 6.8899, 7.1118, 7.5469, 7.9735,
01321 7.9511, 8.3014, 8.6418, 8.4757, 8.8256, 9.2294, 9.6923, 10.033,
01322 10.842, 11.851, 11.78, 8.8435, 9.1381, 9.5956, 10.076, 10.629,
01323 11.22, 11.883, 12.69, 13.163, 13.974, 14.846, 16.027, 17.053,
01324 18.148, 19.715, 20.907, 22.163, 23.956, 25.235, 26.566, 27.94,
01325 29.576, 30.956, 32.432, 35.337, 39.911, 41.128, 42.625, 44.386,
01326 46.369, 48.619, 51.031, 53.674, 56.825, 59.921, 63.286, 66.929,
01327 70.859, 75.081, 79.618, 84.513, 89.739, 95.335, 101.35, 107.76,
01328 114.63, 121.98, 129.87, 138.3, 147.34, 157.04, 167.56, 178.67,
01329 190.61, 203.43, 217.19, 231.99, 247.88, 264.98, 283.37, 303.17,
01330 324.49, 347.47, 372.25, 398.98, 427.85, 459.06, 492.8, 529.31,
01331 568.89, 611.79, 658.35, 708.91, 763.87, 823.65, 888.72, 959.58,
01332 1036.8, 1121.8, 1213.9, 1314.3, 1423.8, 1543, 1672.8, 1813.4,
01333 1966.1, 2131.4, 2309.5, 2499.3, 2705, 2925.7, 3161.6, 3411.3,
01334 3611.5, 3889.2, 4191.1, 4519.3, 4877.9, 5272.9, 5712.9, 6142.7,
01335 6719.6, 7385, 8145, 8977.7, 9831.9, 10827, 11934, 13063,
01336 14434, 15878, 17591, 19435, 21510, 23835, 26835, 29740,
01337 32878, 36305, 39830, 43273, 46931, 50499, 49586, 51598,
01338 53429, 54619, 55081, 55102, 54485, 53487, 52042, 42689,
01339 42607, 44020, 47994, 54169, 53916, 55808, 56642, 46049,
01340 44243, 32929, 30658, 21963, 20835, 15962, 13679, 17652,
01341 19680, 22388, 25625, 29184, 32520, 35720, 38414, 40523,
01342 49228, 48173, 45678, 41768, 37600, 41313, 42654, 44465,
01343 55736, 56630, 65409, 63308, 66572, 61845, 60379, 56777,
01344 51920, 46601, 41367, 36529, 32219, 28470, 25192, 22362,
01345 19907, 17772, 15907, 14273, 12835, 11567, 10445, 9450.2,
01346 8565.1, 7776, 7070.8, 6439.2, 5872.3, 5362.4, 4903, 4488.3,
01347 4113.4, 3773.8, 3465.8, 3186.1, 2931.7, 2700.1, 2488.8, 2296,
01348 2119.8, 1958.6, 1810.9, 1675.6, 1551.4, 1437.3, 1332.4, 1236,
01349 1147.2, 1065.3, 989.86, 920.22, 855.91, 796.48, 741.53, 690.69,
01350 643.62, 600.02, 559.6, 522.13, 487.35, 455.06, 425.08, 397.21,
01351 371.3, 347.2, 324.78, 303.9, 284.46, 266.34, 249.45, 233.7,
01352 219.01, 205.3, 192.5, 180.55, 169.38, 158.95, 149.2, 140.07,
01353 131.54, 123.56, 116.09, 109.09, 102.54, 96.405, 90.655, 85.266,
01354 80.213, 75.475, 71.031, 66.861, 62.948, 59.275, 55.827, 52.587,
01355 49.544, 46.686, 43.998, 41.473, 39.099, 36.867, 34.768, 32.795,
01356 30.939, 29.192, 27.546, 25.998, 24.539, 23.164, 21.869, 20.65,
01357 19.501, 18.419, 17.399, 16.438, 15.532, 14.678, 13.874, 13.115,
01358 12.4, 11.726, 11.088, 10.488, 9.921, 9.3846, 8.8784, 8.3996,
01359 7.9469, 7.5197, 7.1174, 6.738, 6.379, 6.0409, 5.7213, 5.419,
01360 5.1327, 4.8611, 4.6046, 4.3617, 4.1316, 3.9138, 3.7077, 3.5125,
01361 3.3281, 3.1536, 2.9885, 2.8323, 2.6846, 2.5447, 2.4124, 2.2871,
01362 2.1686, 2.0564, 1.9501, 1.8495, 1.7543, 1.6641, 1.5787, 1.4978,
01363 1.4212, 1.3486, 1.2799, 1.2147, 1.1529, 1.0943, 1.0388, .98602,
01364 .93596, .8886, .84352, .80078, .76029, .722, .68585, .65161,
01365 .61901, .58808, .55854, .53044, .5039, .47853, .45459, .43173,
01366 .41008, .38965, .37021, .35186, .33444, .31797, .30234, .28758,
01367 .2736, .26036, .24764, .2357, .22431, .21342, .20295, .19288,
01368 .18334, .17444, .166, .15815, .15072, .14348, .13674, .13015,
01369 .12399, .11807, .11231, .10689, .10164, .096696, .091955,
01370 .087476, .083183, .079113, .075229, .071536, .068026, .064698,
01371 .06154, .058544, .055699, .052997, .050431, .047993, .045676,
01372 .043475, .041382, .039392, .037501, .035702, .033991, .032364,
01373 .030817, .029345, .027945, .026613, .025345, .024139, .022991,
01374 .021899, .02086, .019871, .018929, .018033, .01718, .016368,
01375 .015595, .014859, .014158, .013491, .012856, .012251, .011675,
01376 .011126, .010604, .010107, .0096331, .009182, .0087523, .0083431,
01377 .0079533, .0075821, .0072284, .0068915, .0065706, .0062649,
01378 .0059737, .0056963, .005432, .0051802, .0049404, .0047118,
01379 .0044941, .0042867, .0040891, .0039009, .0037216, .0035507,
01380 .003388, .0032329, .0030852, .0029445, .0028105, .0026829,

```

```
01381 .0025613, .0024455, .0023353, .0022303, .0021304, .0020353,
01382 .0019448, .0018587, .0017767, .0016988, .0016247, .0015543,
01383 .0014874, .0014238, .0013635, .0013062, .0012519, .0012005,
01384 .0011517, .0011057, .0010621, .001021, 9.8233e-4, 9.4589e-4,
01385 9.1167e-4, 8.7961e-4, 8.4964e-4, 8.2173e-4, 7.9582e-4, 7.7189e-4,
01386 7.499e-4, 7.2983e-4, 7.1167e-4, 6.9542e-4, 6.8108e-4, 6.6866e-4,
01387 6.5819e-4, 6.4971e-4, 6.4328e-4, 6.3895e-4, 6.3681e-4, 6.3697e-4,
01388 6.3956e-4, 6.4472e-4, 6.5266e-4, 6.6359e-4, 6.778e-4, 6.9563e-4,
01389 7.1749e-4, 7.4392e-4, 7.7556e-4, 8.1028e-4, 8.4994e-4, 8.8709e-4,
01390 9.3413e-4, 9.6953e-4, .0010202, .0010738, .0010976, .0011507,
01391 .0011686, .0012264, .001291, .0013346, .0014246, .0015293,
01392 .0016359, .0017824, .0019255, .0020854, .002247, .0024148,
01393 .0026199, .0027523, .0029704, .0030702, .0033047, .0035013,
01394 .0037576, .0040275, .0043089, .0046927, .0049307, .0053486,
01395 .0053809, .0056699, .0059325, .0055488, .005634, .0056392,
01396 .004946, .0048855, .0048208, .0044386, .0045498, .0046377,
01397 .0048939, .0052396, .0057324, .0060859, .0066906, .0071148,
01398 .0077224, .0082687, .008769, .0084471, .008572, .0087729,
01399 .008775, .0090742, .0080704, .0080288, .0085747, .0086087,
01400 .0086408, .0088752, .0089381, .0089757, .0093532, .0092824,
01401 .0092566, .0092645, .0092735, .009342, .0095806, .0097991,
01402 .010213, .010611, .011129, .011756, .013237, .01412, .015034,
01403 .015936, .01682, .018597, .019315, .019995, .020658, .021289,
01404 .022363, .022996, .023716, .024512, .025434, .026067, .027118,
01405 .028396, .029865, .031442, .033253, .03525, .037296, .039701,
01406 .042356, .045154, .048059, .051294, .054893, .058636, .061407,
01407 .065172, .068974, .072676, .073379, .076547, .079556, .079134,
01408 .082308, .085739, .090192, .09359, .099599, .10669, .11496,
01409 .1244, .13512, .14752, .14494, .15647, .1668, .17863, .19029,
01410 .20124, .20254, .21179, .21982, .21625, .22364, .23405, .23382,
01411 .2434, .25708, .26406, .27621, .28909, .30395, .31717, .33271,
01412 .3496, .36765, .38774, .40949, .446, .46985, .49846, .5287, .562,
01413 .59841, .64598, .68834, .7327, .78978, .8373, .88708, .94744,
01414 1.0006, 1.0574, 1.1215, 1.1856, 1.2546, 1.3292, 1.4107, 1.4974,
01415 1.5913, 1.6931, 1.8028, 1.9212, 2.0492, 2.1874, 2.3365, 2.4978,
01416 2.6718, 2.8588, 3.062, 3.2818, 3.5188, 3.7752, 4.0527, 4.3542,
01417 4.6782, 5.0312, 5.4123, 5.8246, 6.2639, 6.7435, 7.2636, 7.8064,
01418 8.4091, 9.0696, 9.7677, 10.548, 11.4, 12.309, 13.324, 14.284,
01419 15.445, 16.687, 18.019, 19.403, 20.847, 22.366, 23.925, 25.537,
01420 27.213, 28.069, 29.864, 31.829, 33.988, 35.856, 38.829, 42.321,
01421 46.319, 50.606, 55.126, 59.126, 64.162, 68.708, 74.615, 81.176,
01422 87.739, 95.494, 103.83, 113.38, 123.99, 135.8, 148.7, 162.58,
01423 176.32, 192.6, 211.47, 232.7, 252.64, 277.41, 305.38, 333.44,
01424 366.42, 402.66, 442.14, 484.53, 526.42, 568.15, 558.78, 582.6,
01425 600.98, 613.94, 619.44, 618.24, 609.84, 595.96, 484.86, 475.59,
01426 478.49, 501.56, 552.19, 628.44, 630.39, 658.92, 671.96, 562.7,
01427 545.88, 423.43, 400.14, 306.59, 294.13, 246.8, 226.51, 278.21,
01428 314.39, 347.22, 389.13, 433.16, 477.48, 521.67, 560.54, 683.6,
01429 696.37, 695.91, 683.1, 658.24, 634.89, 698.85, 742.87, 796.66,
01430 954.49, 1009.5, 1150.5, 1179.1, 1267.9, 1272.4, 1312.7, 1330.4,
01431 1331.6, 1315.8, 1308.3, 1293.3, 1274.6, 1249.5, 1213.2, 1172.1,
01432 1124.4, 930.33, 893.36, 871.27, 883.54, 940.76, 1036., 1025.6,
01433 1053.1, 914.51, 894.15, 865.03, 670.63, 508.41, 475.15, 370.85,
01434 361.06, 319.38, 312.75, 331.87, 367.13, 415., 467.94, 525.49,
01435 578.41, 624.66, 794.82, 796.97, 780.29, 736.49, 670.18, 603.75,
01436 659.67, 679.8, 857.12, 884.05, 900.65, 1046.1, 1141.9, 1083.,
01437 1089.2, 1e3, 947.08, 872.31, 787.91, 704.75, 624.93, 553.68,
01438 489.91, 434.21, 385.64, 343.3, 306.42, 274.18, 245.94, 221.11,
01439 199.23, 179.88, 162.73, 147.48, 133.88, 121.73, 110.86, 101.1,
01440 92.323, 84.417, 77.281, 70.831, 64.991, 59.694, 54.884, 50.509,
01441 46.526, 42.893, 39.58, 36.549, 33.776, 31.236, 28.907, 26.77,
01442 24.805, 23., 21.339, 19.81, 18.404, 17.105, 15.909, 14.801,
01443 13.778, 12.83, 11.954, 11.142, 10.389, 9.691, 9.0434, 8.4423,
01444 7.8842, 7.3657, 6.8838, 6.4357, 6.0189, 5.6308, 5.2696, 4.9332,
01445 4.6198, 4.3277, 4.0553, 3.8012, 3.5639, 3.3424, 3.1355, 2.9422,
01446 2.7614, 2.5924, 2.4343, 2.2864, 2.148, 2.0184, 1.8971, 1.7835,
01447 1.677, 1.5773, 1.4838, 1.3961, 1.3139, 1.2369, 1.1645, 1.0966,
01448 1.0329, .97309, .91686, .86406, .81439, .76767, .72381, .68252,
01449 .64359, .60695, .57247, .54008, .50957, .48092, .45401, .42862,
01450 .40465, .38202, .36072, .34052, .3216, .30386, .28711, .27135,
01451 .25651, .24252, .2293, .21689, .20517, .19416, .18381, .17396,
01452 .16469
01453 };
```

```
01454
01455 static double co2230[2001] = { 2.743e-5, 2.8815e-5, 3.027e-5, 3.1798e-5,
01456 3.3405e-5, 3.5094e-5, 3.6869e-5, 3.8734e-5, 4.0694e-5, 4.2754e-5,
01457 4.492e-5, 4.7196e-5, 4.9588e-5, 5.2103e-5, 5.4747e-5, 5.7525e-5,
01458 6.0446e-5, 6.3516e-5, 6.6744e-5, 7.0137e-5, 7.3704e-5, 7.7455e-5,
01459 8.1397e-5, 8.5543e-5, 8.9901e-5, 9.4484e-5, 9.9302e-5, 1.0437e-4,
01460 1.097e-4, 1.153e-4, 1.2119e-4, 1.2738e-4, 1.3389e-4, 1.4074e-4,
01461 1.4795e-4, 1.5552e-4, 1.6349e-4, 1.7187e-4, 1.8068e-4, 1.8995e-4,
01462 1.997e-4, 2.0996e-4, 2.2075e-4, 2.321e-4, 2.4403e-4, 2.5659e-4,
01463 2.698e-4, 2.837e-4, 2.9832e-4, 3.137e-4, 3.2988e-4, 3.4691e-4,
01464 3.6483e-4, 3.8368e-4, 4.0351e-4, 4.2439e-4, 4.4635e-4, 4.6947e-4,
01465 4.9379e-4, 5.1939e-4, 5.4633e-4, 5.7468e-4, 6.0452e-4, 6.3593e-4,
01466 6.69e-4, 7.038e-4, 7.4043e-4, 7.79e-4, 8.1959e-4, 8.6233e-4,
01467 9.0732e-4, 9.5469e-4, .0010046, .0010571, .0011124, .0011706,
```

```
01468 .0012319, .0012964, .0013644, .001436, .0015114, .0015908,
01469 .0016745, .0017625, .0018553, .0019531, .002056, .0021645,
01470 .0022788, .0023992, .002526, .0026596, .0028004, .0029488,
01471 .0031052, .0032699, .0034436, .0036265, .0038194, .0040227,
01472 .0042369, .0044628, .0047008, .0049518, .0052164, .0054953,
01473 .0057894, .0060995, .0064265, .0067713, .007135, .0075184,
01474 .0079228, .0083494, .0087993, .0092738, .0097745, .010303,
01475 .01086, .011448, .012068, .012722, .013413, .014142, .014911,
01476 .015723, .01658, .017484, .018439, .019447, .020511, .021635,
01477 .022821, .024074, .025397, .026794, .02827, .029829, .031475,
01478 .033215, .035052, .036994, .039045, .041213, .043504, .045926,
01479 .048485, .05119, .05405, .057074, .060271, .063651, .067225,
01480 .071006, .075004, .079233, .083708, .088441, .093449, .098749,
01481 .10436, .11029, .11657, .12322, .13026, .13772, .14561, .15397,
01482 .16282, .1722, .18214, .19266, .20381, .21563, .22816, .24143,
01483 .2555, .27043, .28625, .30303, .32082, .3397, .35972, .38097,
01484 .40352, .42746, .45286, .47983, .50847, .53888, .57119, .6055,
01485 .64196, .6807, .72187, .76564, .81217, .86165, .91427, .97025,
01486 1.0298, 1.0932, 1.1606, 1.2324, 1.3088, 1.3902, 1.477, 1.5693,
01487 1.6678, 1.7727, 1.8845, 2.0038, 2.131, 2.2666, 2.4114, 2.5659,
01488 2.7309, 2.907, 3.0951, 3.2961, 3.5109, 3.7405, 3.986, 4.2485,
01489 4.5293, 4.8299, 5.1516, 5.4961, 5.8651, 6.2605, 6.6842, 7.1385,
01490 7.6256, 8.1481, 8.7089, 9.3109, 9.9573, 10.652, 11.398, 12.2,
01491 13.063, 13.992, 14.99, 16.064, 17.222, 18.469, 19.813, 21.263,
01492 22.828, 24.516, 26.34, 28.31, 30.437, 32.738, 35.226, 37.914,
01493 40.824, 43.974, 47.377, 51.061, 55.011, 59.299, 63.961, 69.013,
01494 74.492, 80.444, 86.919, 93.836, 101.23, 109.25, 117.98, 127.47,
01495 137.81, 149.07, 161.35, 174.75, 189.42, 205.49, 223.02, 242.26,
01496 263.45, 286.75, 311.94, 340.01, 370.86, 404.92, 440.44, 480.27,
01497 525.17, 574.71, 626.22, 686.8, 754.38, 827.07, 913.38, 1011.7,
01498 1121.5, 1161.6, 1289.5, 1432.2, 1595.4, 1777., 1983.3, 2216.1,
01499 2485.7, 2788.3, 3101.5, 3481., 3902.1, 4257.1, 4740., 5272.8,
01500 5457.9, 5946.2, 6505.3, 6668.4, 7302.4, 8061.6, 9015.8, 9908.3,
01501 11613., 13956., 3249.6, 3243., 2901.5, 2841.3, 2729.6, 2558.2,
01502 1797.8, 1583.2, 1386., 1233.5, 787.74, 701.46, 761.66, 767.21,
01503 722.83, 1180.6, 1332.1, 1461.6, 2032.9, 2166., 2255.9, 2294.7,
01504 2587.2, 2396.5, 2122.4, 12553., 10784., 9832.5, 8827.3, 8029.1,
01505 7377.9, 7347.1, 6783.8, 6239.1, 5721.1, 5503., 4975.1, 4477.8,
01506 4021.3, 3676.8, 3275.3, 2914.9, 2597.4, 2328.2, 2075.4, 1857.6,
01507 1663.6, 1493.3, 1343.8, 1213.3, 1095.6, 1066.5, 958.91, 865.15,
01508 783.31, 714.35, 650.77, 593.98, 546.2, 499.9, 457.87, 421.75,
01509 387.61, 355.25, 326.62, 299.7, 275.21, 253.17, 232.83, 214.31,
01510 197.5, 182.08, 167.98, 155.12, 143.32, 132.5, 122.58, 113.48,
01511 105.11, 97.415, 90.182, 83.463, 77.281, 71.587, 66.341, 61.493,
01512 57.014, 53.062, 49.21, 45.663, 42.38, 39.348, 36.547, 33.967,
01513 31.573, 29.357, 27.314, 25.415, 23.658, 22.03, 20.524, 19.125,
01514 17.829, 16.627, 15.511, 14.476, 13.514, 12.618, 11.786, 11.013,
01515 10.294, 9.6246, 9.0018, 8.4218, 7.8816, 7.3783, 6.9092, 6.4719,
01516 6.0641, 5.6838, 5.3289, 4.998, 4.6893, 4.4014, 4.1325, 3.8813,
01517 3.6469, 3.4283, 3.2241, 3.035, 2.8576, 2.6922, 2.5348, 2.3896,
01518 2.2535, 2.1258, 2.0059, 1.8929, 1.7862, 1.6854, 1.5898, 1.4992,
01519 1.4017, 1.3218, 1.2479, 1.1809, 1.1215, 1.0693, 1.0116, .96016,
01520 .9105, .84859, .80105, .74381, .69982, .65127, .60899, .57843,
01521 .54592, .51792, .49336, .47155, .45201, .43426, .41807, .40303,
01522 .38876, .3863, .37098, .35492, .33801, .32032, .30341, .29874,
01523 .29193, .28689, .29584, .29155, .29826, .29195, .29287, .2904,
01524 .28199, .27709, .27162, .26622, .26133, .25676, .25235, .23137,
01525 .22365, .21519, .20597, .19636, .18699, .16485, .16262, .16643,
01526 .17542, .18198, .18631, .16759, .16338, .13505, .1267, .10053,
01527 .092554, .074093, .062159, .055523, .054849, .05401, .05528,
01528 .058982, .07952, .08647, .093244, .099285, .10393, .10661,
01529 .12072, .11417, .10396, .093265, .089137, .088909, .10902,
01530 .11277, .13625, .13565, .14907, .14167, .1428, .13744, .12768,
01531 .11382, .10244, .091686, .08109, .071739, .063616, .056579,
01532 .050504, .045251, .040689, .036715, .033237, .030181, .027488,
01533 .025107, .022998, .021125, .01946, .017979, .016661, .015489,
01534 .014448, .013526, .012712, .011998, .011375, .010839, .010384,
01535 .010007, .0097053, .0094783, .0093257, .0092489, .0092504,
01536 .0093346, .0095077, .0097676, .01012, .01058, .011157, .011844,
01537 .012672, .013665, .014766, .015999, .017509, .018972, .020444,
01538 .022311, .023742, .0249, .025599, .026981, .026462, .025143,
01539 .025066, .022814, .020458, .020026, .019142, .020189, .022371,
01540 .024163, .023728, .02199, .019506, .018591, .015576, .012784,
01541 .011744, .0094777, .0079148, .0070652, .006986, .0071758,
01542 .008086, .0098025, .01087, .013609, .016764, .018137, .021061,
01543 .023498, .023576, .023965, .022828, .021519, .021283, .023364,
01544 .026457, .029782, .030856, .033486, .035515, .035543, .036558,
01545 .037198, .037472, .037045, .037284, .03777, .038085, .038366,
01546 .038526, .038282, .038915, .037697, .035667, .032941, .031959,
01547 .028692, .025918, .024596, .025592, .027873, .028935, .02984,
01548 .028148, .025305, .021912, .020454, .016732, .013357, .01205,
01549 .009731, .0079881, .0077704, .0074387, .0083895, .0096776,
01550 .010326, .01293, .015955, .019247, .020145, .02267, .024231,
01551 .024184, .022131, .019784, .01955, .01971, .022119, .025116,
01552 .027978, .028107, .029808, .030701, .029164, .028551, .027286,
01553 .024946, .023259, .020982, .019221, .017471, .015643, .014074,
01554 .01261, .011301, .010116, .0090582, .0081036, .0072542, .0065034,
```

01555 .0058436, .0052571, .0047321, .0042697, .0038607, .0034977,  
01556 .0031747, .0028864, .0026284, .002397, .002189, .0020017,  
01557 .0018326, .0016798, .0015414, .0014159, .0013019, .0011983,  
01558 .0011039, .0010177, 9.391e-4, 8.6717e-4, 8.0131e-4, 7.4093e-4,  
01559 6.8553e-4, 6.3464e-4, 5.8787e-4, 5.4487e-4, 5.0533e-4, 4.69e-4,  
01560 4.3556e-4, 4.0474e-4, 3.7629e-4, 3.5e-4, 3.2569e-4, 3.032e-4,  
01561 2.8239e-4, 2.6314e-4, 2.4535e-4, 2.2891e-4, 2.1374e-4, 1.9975e-4,  
01562 1.8685e-4, 1.7498e-4, 1.6406e-4, 1.5401e-4, 1.4479e-4, 1.3633e-4,  
01563 1.2858e-4, 1.2148e-4, 1.1499e-4, 1.0907e-4, 1.0369e-4, 9.8791e-5,  
01564 9.4359e-5, 9.0359e-5, 8.6766e-5, 8.3555e-5, 8.0703e-5, 7.8192e-5,  
01565 7.6003e-5, 7.4119e-5, 7.2528e-5, 7.1216e-5, 7.0171e-5, 6.9385e-5,  
01566 6.8848e-5, 6.8554e-5, 6.8496e-5, 6.8669e-5, 6.9069e-5, 6.9694e-5,  
01567 7.054e-5, 7.1608e-5, 7.2896e-5, 7.4406e-5, 7.6139e-5, 7.8097e-5,  
01568 8.0283e-5, 8.2702e-5, 8.5357e-5, 8.8255e-5, 9.1402e-5, 9.4806e-5,  
01569 9.8473e-5, 1.0241e-4, 1.0664e-4, 1.1115e-4, 1.1598e-4, 1.2112e-4,  
01570 1.2659e-4, 1.3241e-4, 1.3859e-4, 1.4515e-4, 1.521e-4, 1.5947e-4,  
01571 1.6728e-4, 1.7555e-4, 1.8429e-4, 1.9355e-4, 2.0334e-4, 2.1369e-4,  
01572 2.2463e-4, 2.3619e-4, 2.4841e-4, 2.6132e-4, 2.7497e-4, 2.8938e-4,  
01573 3.0462e-4, 3.2071e-4, 3.3771e-4, 3.5567e-4, 3.7465e-4, 3.947e-4,  
01574 4.1588e-4, 4.3828e-4, 4.6194e-4, 4.8695e-4, 5.1338e-4, 5.4133e-4,  
01575 5.7087e-4, 6.0211e-4, 6.3515e-4, 6.701e-4, 7.0706e-4, 7.4617e-4,  
01576 7.8756e-4, 8.3136e-4, 8.7772e-4, 9.2681e-4, 9.788e-4, .0010339,  
01577 .0010922, .001154, .0012195, .0012889, .0013626, .0014407,  
01578 .0015235, .0016114, .0017048, .0018038, .001909, .0020207,  
01579 .0021395, .0022657, .0023998, .0025426, .0026944, .002856,  
01580 .0030281, .0032114, .0034068, .003615, .0038371, .004074,  
01581 .004327, .0045971, .0048857, .0051942, .0055239, .0058766,  
01582 .0062538, .0066573, .0070891, .007551, .0080455, .0085747,  
01583 .0091412, .0097481, .010397, .011092, .011837, .012638, .013495,  
01584 .014415, .01541, .016475, .017621, .018857, .020175, .02162,  
01585 .023185, .024876, .02672, .028732, .030916, .033319, .035939,  
01586 .038736, .041847, .04524, .048715, .052678, .056977, .061203,  
01587 .066184, .07164, .076952, .083477, .090674, .098049, .10697,  
01588 .1169, .1277, .14011, .15323, .1684, .18601, .20626, .22831,  
01589 .25417, .28407, .31405, .34957, .38823, .41923, .46026, .50409,  
01590 .51227, .54805, .57976, .53818, .55056, .557, .46741, .46403,  
01591 .4636, .42265, .45166, .49852, .56663, .34306, .17779, .17697,  
01592 .18346, .19129, .20014, .21778, .23604, .25649, .28676, .31238,  
01593 .33856, .39998, .4288, .46568, .56654, .60786, .64473, .76466,  
01594 .7897, .80778, .86443, .85736, .84798, .84157, 1.1385, 1.2446,  
01595 1.1923, 1.1552, 1.1338, 1.1266, 1.1292, 1.1431, 1.1683, 1.2059,  
01596 1.2521, 1.3069, 1.3712, 1.4471, 1.5275, 1.6165, 1.7145, 1.8189,  
01597 1.9359, 2.065, 2.2007, 2.3591, 2.5362, 2.7346, 2.9515, 3.2021,  
01598 3.4851, 3.7935, 4.0694, 4.4463, 4.807, 5.2443, 5.7178, 6.2231,  
01599 6.4796, 6.9461, 7.4099, 7.3652, 7.7182, 8.048, 7.7373, 8.0363,  
01600 8.3855, 8.8044, 9.0257, 9.8574, 10.948, 10.563, 6.8979, 7.0744,  
01601 7.4121, 7.7663, 8.1768, 8.6243, 9.1437, 9.7847, 10.182, 10.849,  
01602 11.572, 12.602, 13.482, 14.431, 15.907, 16.983, 18.11, 19.884,  
01603 21.02, 22.18, 23.355, 24.848, 25.954, 27.13, 30.186, 34.893,  
01604 35.682, 36.755, 38.111, 39.703, 41.58, 43.606, 45.868, 48.573,  
01605 51.298, 54.291, 57.559, 61.116, 64.964, 69.124, 73.628, 78.471,  
01606 83.683, 89.307, 95.341, 101.84, 108.83, 116.36, 124.46, 133.18,  
01607 142.57, 152.79, 163.69, 175.43, 188.11, 201.79, 216.55, 232.51,  
01608 249.74, 268.38, 288.54, 310.35, 333.97, 359.55, 387.26, 417.3,  
01609 449.88, 485.2, 523.54, 565.14, 610.28, 659.31, 712.56, 770.43,  
01610 833.36, 901.82, 976.36, 1057.6, 1146.8, 1243.8, 1350., 1466.3,  
01611 1593.6, 1732.7, 1884.1, 2049.1, 2228.2, 2421.9, 2629.4, 2853.7,  
01612 3094.4, 3351.1, 3622.3, 3829.8, 4123.1, 4438.3, 4777.2, 5144.1,  
01613 5545.4, 5990.5, 6404.5, 6996.8, 7687.6, 8482.9, 9349.4, 10203.,  
01614 11223., 12358., 13493., 14916., 16416., 18236., 20222., 22501.,  
01615 25102., 28358., 31707., 35404., 39538., 43911., 48391., 53193.,  
01616 58028., 58082., 61276., 64193., 66294., 67480., 67921., 67423.,  
01617 66254., 64341., 51737., 51420., 53072., 58145., 66195., 65358.,  
01618 67377., 67869., 53509., 50553., 35737., 32425., 21704., 19974.,  
01619 14457., 12142., 16798., 19489., 23049., 27270., 31910., 36457.,  
01620 40877., 44748., 47876., 59793., 58626., 55454., 50337., 44893.,  
01621 50228., 52216., 54747., 69541., 70455., 81014., 77694., 80533.,  
01622 73953., 70927., 65539., 59002., 52281., 45953., 40292., 35360.,  
01623 31124., 27478., 24346., 21647., 19308., 17271., 15491., 13927.,  
01624 12550., 11331., 10250., 9288.8, 8431.4, 7664.9, 6978.3, 6361.8,  
01625 5807.4, 5307.7, 4856.8, 4449., 4079.8, 3744.9, 3440.8, 3164.2,  
01626 2912.3, 2682.7, 2473., 2281.4, 2106., 1945.3, 1797.9, 1662.5,  
01627 1538.1, 1423.6, 1318.1, 1221., 1131.5, 1049., 972.99, 902.87,  
01628 838.01, 777.95, 722.2, 670.44, 622.35, 577.68, 536.21, 497.76,  
01629 462.12, 429.13, 398.61, 370.39, 344.29, 320.16, 297.85, 277.2,  
01630 258.08, 240.38, 223.97, 208.77, 194.66, 181.58, 169.43, 158.15,  
01631 147.67, 137.92, 128.86, 120.44, 112.6, 105.3, 98.499, 92.166,  
01632 86.264, 80.763, 75.632, 70.846, 66.381, 62.213, 58.321, 54.685,  
01633 51.288, 48.114, 45.145, 42.368, 39.772, 37.341, 35.065, 32.937,  
01634 30.943, 29.077, 27.33, 25.693, 24.158, 22.717, 21.367, 20.099,  
01635 18.909, 17.792, 16.744, 15.761, 14.838, 13.971, 13.157, 12.393,  
01636 11.676, 11.003, 10.369, 9.775, 9.2165, 8.6902, 8.1963, 7.7314,  
01637 7.2923, 6.8794, 6.4898, 6.122, 5.7764, 5.4525, 5.1484, 4.8611,  
01638 4.5918, 4.3379, 4.0982, 3.8716, 3.6567, 3.4545, 3.2634, 3.0828,  
01639 2.9122, 2.7512, 2.5993, 2.4561, 2.3211, 2.1938, 2.0737, 1.9603,  
01640 1.8534, 1.7525, 1.6572, 1.5673, 1.4824, 1.4022, 1.3265, 1.2551,  
01641 1.1876, 1.1239, 1.0637, 1.0069, .9532, .90248, .85454, .80921,

```
01642 .76631, .72569, .6872, .65072, .61635, .5836, .55261, .52336,
01643 .49581, .46998, .44559, .42236, .40036, .37929, .35924, .34043,
01644 .32238, .30547, .28931, .27405, .25975, .24616, .23341, .22133,
01645 .20997, .19924, .18917, .17967, .17075, .16211, .15411, .14646,
01646 .13912, .13201, .12509, .11857, .11261, .10698, .10186, .097039,
01647 .092236, .087844, .083443, .07938, .075452, .071564, .067931,
01648 .064389, .061078, .057901, .054921, .052061, .049364, .046789,
01649 .04435, .042044, .039866, .037808, .035863, .034023, .032282,
01650 .030634, .029073, .027595, .026194, .024866, .023608, .022415,
01651 .021283, .02021, .019193, .018228, .017312, .016443, .015619,
01652 .014837, .014094, .01339, .012721, .012086, .011483, .010911,
01653 .010368, .009852, .0093623, .0088972, .0084556, .0080362,
01654 .0076379, .0072596, .0069003, .006559, .0062349, .0059269,
01655 .0056344, .0053565, .0050925, .0048417, .0046034, .004377,
01656 .0041618, .0039575, .0037633, .0035788, .0034034, .0032368,
01657 .0030785, .002928, .0027851, .0026492, .0025201, .0023975,
01658 .0022809, .0021701, .0020649, .0019649, .0018699, .0017796,
01659 .0016938, .0016122, .0015348, .0014612, .0013913, .001325,
01660 .0012619, .0012021, .0011452, .0010913, .0010401, 9.9149e-4,
01661 9.454e-4, 9.0169e-4, 8.6024e-4, 8.2097e-4, 7.8377e-4, 7.4854e-4,
01662 7.1522e-4, 6.8371e-4, 6.5393e-4, 6.2582e-4, 5.9932e-4, 5.7435e-4,
01663 5.5087e-4, 5.2882e-4, 5.0814e-4, 4.8881e-4, 4.7076e-4, 4.5398e-4,
01664 4.3843e-4, 4.2407e-4, 4.109e-4, 3.9888e-4, 3.88e-4, 3.7826e-4,
01665 3.6963e-4, 3.6213e-4, 3.5575e-4, 3.505e-4, 3.464e-4, 3.4346e-4,
01666 3.4173e-4, 3.4125e-4, 3.4206e-4, 3.4424e-4, 3.4787e-4, 3.5303e-4,
01667 3.5986e-4, 3.6847e-4, 3.7903e-4, 3.9174e-4, 4.0681e-4, 4.2455e-4,
01668 4.4527e-4, 4.6942e-4, 4.9637e-4, 5.2698e-4, 5.5808e-4, 5.9514e-4,
01669 6.2757e-4, 6.689e-4, 7.1298e-4, 7.3955e-4, 7.8403e-4, 8.0449e-4,
01670 8.5131e-4, 9.0256e-4, 9.3692e-4, .0010051, .0010846, .0011678,
01671 .001282, .0014016, .0015355, .0016764, .0018272, .0020055,
01672 .0021455, .0023421, .0024615, .0026786, .0028787, .0031259,
01673 .0034046, .0036985, .0040917, .0043902, .0048349, .0049531,
01674 .0052989, .0056148, .0052452, .0053357, .005333, .0045069,
01675 .0043851, .004253, .003738, .0038084, .0039013, .0041505,
01676 .0045372, .0050569, .0054507, .0061267, .0066122, .0072449,
01677 .0078012, .0082651, .0076538, .0076573, .0076806, .0075227,
01678 .0076269, .0063758, .006254, .0067749, .0067909, .0068231,
01679 .0072143, .0072762, .0072954, .007679, .0075107, .0073658,
01680 .0072441, .0071074, .0070378, .007176, .0072472, .0075844,
01681 .0079291, .008412, .0090165, .010688, .011535, .012375, .013166,
01682 .013895, .015567, .016011, .016392, .016737, .017043, .017731,
01683 .018031, .018419, .018877, .019474, .019868, .020604, .021538,
01684 .022653, .023869, .025288, .026879, .028547, .030524, .03274,
01685 .035132, .03769, .040567, .043793, .047188, .049962, .053542,
01686 .057205, .060776, .061489, .064419, .067124, .065945, .068487,
01687 .071209, .074783, .077039, .082444, .08902, .09692, .10617,
01688 .11687, .12952, .12362, .13498, .14412, .15492, .16519, .1744,
01689 .17096, .17714, .18208, .17363, .17813, .18564, .18295, .19045,
01690 .20252, .20815, .21844, .22929, .24229, .25321, .26588, .2797,
01691 .29465, .31136, .32961, .36529, .38486, .41027, .43694, .4667,
01692 .49943, .54542, .58348, .62303, .67633, .71755, .76054, .81371,
01693 .85934, .90841, .96438, 1.0207, 1.0821, 1.1491, 1.2226, 1.3018,
01694 1.388, 1.4818, 1.5835, 1.6939, 1.8137, 1.9435, 2.0843, 2.237,
01695 2.4026, 2.5818, 2.7767, 2.9885, 3.2182, 3.4679, 3.7391, 4.0349,
01696 4.3554, 4.7053, 5.0849, 5.4986, 5.9436, 6.4294, 6.9598, 7.5203,
01697 8.143, 8.8253, 9.5568, 10.371, 11.267, 12.233, 13.31, 14.357,
01698 15.598, 16.93, 18.358, 19.849, 21.408, 23.04, 24.706, 26.409,
01699 28.153, 28.795, 30.549, 32.43, 34.49, 36.027, 38.955, 42.465,
01700 46.565, 50.875, 55.378, 59.002, 63.882, 67.949, 73.693, 80.095,
01701 86.403, 94.264, 102.65, 112.37, 123.3, 135.54, 149.14, 163.83,
01702 179.17, 196.89, 217.91, 240.94, 264.13, 292.39, 324.83, 358.21,
01703 397.16, 440.5, 488.6, 541.04, 595.3, 650.43, 652.03, 688.74,
01704 719.47, 743.54, 757.68, 762.35, 756.43, 741.42, 595.43, 580.97,
01705 580.83, 605.68, 667.88, 764.49, 759.93, 789.12, 798.17, 645.66,
01706 615.65, 455.05, 421.09, 306.45, 289.14, 235.7, 215.52, 274.57,
01707 316.53, 357.73, 409.89, 465.06, 521.84, 579.02, 630.64, 794.46,
01708 813., 813.56, 796.25, 761.57, 727.97, 812.14, 866.75, 932.5,
01709 1132.8, 1194.8, 1362.2, 1387.2, 1482.3, 1479.7, 1517.9, 1533.1,
01710 1534.2, 1523.3, 1522.5, 1515.5, 1505.2, 1486.5, 1454., 1412.,
01711 1358.8, 1107.8, 1060.9, 1033.5, 1048.2, 1122.4, 1248.9, 1227.1,
01712 1255.4, 1058.9, 1020.7, 970.59, 715.24, 512.56, 468.47, 349.3,
01713 338.26, 299.22, 301.26, 332.38, 382.08, 445.49, 515.87, 590.65,
01714 662.3, 726.05, 955.59, 964.11, 945.17, 891.48, 807.11, 720.9,
01715 803.36, 834.46, 1073.9, 1107.1, 1123.6, 1296., 1393.7, 1303.1,
01716 1284.3, 1161.8, 1078.8, 976.13, 868.72, 767.4, 674.72, 593.73,
01717 523.12, 462.24, 409.75, 364.34, 325., 290.73, 260.76, 234.46,
01718 211.28, 190.78, 172.61, 156.44, 142.01, 129.12, 117.57, 107.2,
01719 97.877, 89.47, 81.882, 75.021, 68.807, 63.171, 58.052, 53.396,
01720 49.155, 45.288, 41.759, 38.531, 35.576, 32.868, 30.384, 28.102,
01721 26.003, 24.071, 22.293, 20.655, 19.147, 17.756, 16.476, 15.292,
01722 14.198, 13.183, 12.241, 11.367, 10.554, 9.7989, 9.0978, 8.4475,
01723 7.845, 7.2868, 6.7704, 6.2927, 5.8508, 5.4421, 5.064, 4.714,
01724 4.3902, 4.0902, 3.8121, 3.5543, 3.315, 3.093, 2.8869, 2.6953,
01725 2.5172, 2.3517, 2.1977, 2.0544, 1.9211, 1.7969, 1.6812, 1.5735,
01726 1.4731, 1.3794, 1.2921, 1.2107, 1.1346, 1.0637, .99744, .93554,
01727 .87771, .82368, .77313, .72587, .6816, .64014, .60134, .565,
01728 .53086, .49883, .46881, .44074, .4144, .38979, .36679, .34513,
```



```

01729     .32474, .30552, .28751, .27045, .25458, .23976, .22584, .21278,
01730     .20051, .18899, .17815, .16801, .15846, .14954, .14117, .13328,
01731     .12584
01732 };
01733
01734 /* Get CO2 continuum absorption... */
01735 double xw = nu / 2 + 1;
01736 if (xw >= 1 && xw < 2001) {
01737     int iw = (int) xw;
01738     double dw = xw - iw;
01739     double ew = 1 - dw;
01740     double cw296 = ew * co2296[iw - 1] + dw * co2296[iw];
01741     double cw260 = ew * co2260[iw - 1] + dw * co2260[iw];
01742     double cw230 = ew * co2230[iw - 1] + dw * co2230[iw];
01743     double dt230 = t - 230;
01744     double dt260 = t - 260;
01745     double dt296 = t - 296;
01746     double ctw = dt260 * 5.050505e-4 * dt296 * cw230 - dt230 * 9.259259e-4
01747         * dt296 * cw260 + dt230 * 4.208754e-4 * dt260 * cw296;
01748     return u / NA / 1000 * p / PO * ctw;
01749 } else
01750     return 0;
01751 }

```

**5.19.3.7 ctmh2o()** double ctmh2o (

double nu,

double p,

double t,

double q,

double u )

Compute water vapor continuum (optical depth).

Definition at line 1755 of file [jurassic.c](#).

```

01760     {
01761
01762     static double h2o296[2001] = { .17, .1695, .172, .168, .1687, .1624, .1606,
01763     .1508, .1447, .1344, .1214, .1133, .1009, .09217, .08297, .06989,
01764     .06513, .05469, .05056, .04417, .03779, .03484, .02994, .0272,
01765     .02325, .02063, .01818, .01592, .01405, .01251, .0108, .009647,
01766     .008424, .007519, .006555, .00588, .005136, .004511, .003989,
01767     .003509, .003114, .00274, .002446, .002144, .001895, .001676,
01768     .001486, .001312, .001164, .001031, 9.129e-4, 8.106e-4, 7.213e-4,
01769     6.4e-4, 5.687e-4, 5.063e-4, 4.511e-4, 4.029e-4, 3.596e-4,
01770     3.22e-4, 2.889e-4, 2.597e-4, 2.337e-4, 2.108e-4, 1.907e-4,
01771     1.728e-4, 1.57e-4, 1.43e-4, 1.305e-4, 1.195e-4, 1.097e-4,
01772     1.009e-4, 9.307e-5, 8.604e-5, 7.971e-5, 7.407e-5, 6.896e-5,
01773     6.433e-5, 6.013e-5, 5.631e-5, 5.283e-5, 4.963e-5, 4.669e-5,
01774     4.398e-5, 4.148e-5, 3.917e-5, 3.702e-5, 3.502e-5, 3.316e-5,
01775     3.142e-5, 2.978e-5, 2.825e-5, 2.681e-5, 2.546e-5, 2.419e-5,
01776     2.299e-5, 2.186e-5, 2.079e-5, 1.979e-5, 1.884e-5, 1.795e-5,
01777     1.711e-5, 1.633e-5, 1.559e-5, 1.49e-5, 1.426e-5, 1.367e-5,
01778     1.312e-5, 1.263e-5, 1.218e-5, 1.178e-5, 1.143e-5, 1.112e-5,
01779     1.088e-5, 1.07e-5, 1.057e-5, 1.05e-5, 1.051e-5, 1.059e-5,
01780     1.076e-5, 1.1e-5, 1.133e-5, 1.18e-5, 1.237e-5, 1.308e-5,
01781     1.393e-5, 1.483e-5, 1.614e-5, 1.758e-5, 1.93e-5, 2.123e-5,
01782     2.346e-5, 2.647e-5, 2.93e-5, 3.279e-5, 3.745e-5, 4.152e-5,
01783     4.813e-5, 5.477e-5, 6.203e-5, 7.331e-5, 8.056e-5, 9.882e-5,
01784     1.05e-4, 1.21e-4, 1.341e-4, 1.572e-4, 1.698e-4, 1.968e-4,
01785     2.175e-4, 2.431e-4, 2.735e-4, 2.867e-4, 3.19e-4, 3.371e-4,
01786     3.554e-4, 3.726e-4, 3.837e-4, 3.878e-4, 3.864e-4, 3.858e-4,
01787     3.841e-4, 3.852e-4, 3.815e-4, 3.762e-4, 3.618e-4, 3.579e-4,
01788     3.45e-4, 3.202e-4, 3.018e-4, 2.785e-4, 2.602e-4, 2.416e-4,
01789     2.097e-4, 1.939e-4, 1.689e-4, 1.498e-4, 1.308e-4, 1.17e-4,
01790     1.011e-4, 9.237e-5, 7.909e-5, 7.006e-5, 6.112e-5, 5.401e-5,
01791     4.914e-5, 4.266e-5, 3.963e-5, 3.316e-5, 3.037e-5, 2.598e-5,
01792     2.294e-5, 2.066e-5, 1.813e-5, 1.583e-5, 1.423e-5, 1.247e-5,
01793     1.116e-5, 9.76e-6, 8.596e-6, 7.72e-6, 6.825e-6, 6.108e-6,
01794     5.366e-6, 4.733e-6, 4.229e-6, 3.731e-6, 3.346e-6, 2.972e-6,
01795     2.628e-6, 2.356e-6, 2.102e-6, 1.878e-6, 1.678e-6, 1.507e-6,
01796     1.348e-6, 1.21e-6, 1.089e-6, 9.806e-7, 8.857e-7, 8.004e-7,
01797     7.261e-7, 6.599e-7, 6.005e-7, 5.479e-7, 5.011e-7, 4.595e-7,
01798     4.219e-7, 3.885e-7, 3.583e-7, 3.314e-7, 3.071e-7, 2.852e-7,
01799     2.654e-7, 2.474e-7, 2.311e-7, 2.162e-7, 2.026e-7, 1.902e-7,
01800     1.788e-7, 1.683e-7, 1.587e-7, 1.497e-7, 1.415e-7, 1.338e-7,
01801     1.266e-7, 1.2e-7, 1.138e-7, 1.08e-7, 1.027e-7, 9.764e-8,
01802     9.296e-8, 8.862e-8, 8.458e-8, 8.087e-8, 7.744e-8, 7.429e-8,

```

```

01803    7.145e-8, 6.893e-8, 6.664e-8, 6.468e-8, 6.322e-8, 6.162e-8,
01804    6.07e-8, 5.992e-8, 5.913e-8, 5.841e-8, 5.796e-8, 5.757e-8,
01805    5.746e-8, 5.731e-8, 5.679e-8, 5.577e-8, 5.671e-8, 5.656e-8,
01806    5.594e-8, 5.593e-8, 5.602e-8, 5.62e-8, 5.693e-8, 5.725e-8,
01807    5.858e-8, 6.037e-8, 6.249e-8, 6.535e-8, 6.899e-8, 7.356e-8,
01808    7.918e-8, 8.618e-8, 9.385e-8, 1.039e-7, 1.158e-7, 1.29e-7,
01809    1.437e-7, 1.65e-7, 1.871e-7, 2.121e-7, 2.427e-7, 2.773e-7,
01810    3.247e-7, 3.677e-7, 4.037e-7, 4.776e-7, 5.101e-7, 6.214e-7,
01811    6.936e-7, 7.581e-7, 8.486e-7, 9.355e-7, 9.942e-7, 1.063e-6,
01812    1.123e-6, 1.191e-6, 1.215e-6, 1.247e-6, 1.26e-6, 1.271e-6,
01813    1.284e-6, 1.317e-6, 1.323e-6, 1.349e-6, 1.353e-6, 1.362e-6,
01814    1.344e-6, 1.329e-6, 1.336e-6, 1.327e-6, 1.325e-6, 1.359e-6,
01815    1.374e-6, 1.415e-6, 1.462e-6, 1.526e-6, 1.619e-6, 1.735e-6,
01816    1.863e-6, 2.034e-6, 2.265e-6, 2.482e-6, 2.756e-6, 3.103e-6,
01817    3.466e-6, 3.832e-6, 4.378e-6, 4.913e-6, 5.651e-6, 6.311e-6,
01818    7.169e-6, 8.057e-6, 9.253e-6, 1.047e-5, 1.212e-5, 1.36e-5,
01819    1.569e-5, 1.776e-5, 2.02e-5, 2.281e-5, 2.683e-5, 2.994e-5,
01820    3.488e-5, 3.896e-5, 4.499e-5, 5.175e-5, 6.035e-5, 6.34e-5,
01821    7.281e-5, 7.923e-5, 8.348e-5, 9.631e-5, 1.044e-4, 1.102e-4,
01822    1.176e-4, 1.244e-4, 1.283e-4, 1.326e-4, 1.4e-4, 1.395e-4,
01823    1.387e-4, 1.387e-4, 1.314e-4, 1.241e-4, 1.228e-4, 1.148e-4,
01824    1.086e-4, 1.018e-4, 8.89e-5, 8.316e-5, 7.292e-5, 6.452e-5,
01825    5.625e-5, 5.045e-5, 4.38e-5, 3.762e-5, 3.29e-5, 2.836e-5,
01826    2.485e-5, 2.168e-5, 1.895e-5, 1.659e-5, 1.453e-5, 1.282e-5,
01827    1.132e-5, 1.001e-5, 8.836e-6, 7.804e-6, 6.922e-6, 6.116e-6,
01828    5.429e-6, 4.824e-6, 4.278e-6, 3.788e-6, 3.371e-6, 2.985e-6,
01829    2.649e-6, 2.357e-6, 2.09e-6, 1.858e-6, 1.647e-6, 1.462e-6,
01830    1.299e-6, 1.155e-6, 1.028e-6, 9.142e-7, 8.132e-7, 7.246e-7,
01831    6.451e-7, 5.764e-7, 5.151e-7, 4.603e-7, 4.121e-7, 3.694e-7,
01832    3.318e-7, 2.985e-7, 2.69e-7, 2.428e-7, 2.197e-7, 1.992e-7,
01833    1.81e-7, 1.649e-7, 1.506e-7, 1.378e-7, 1.265e-7, 1.163e-7,
01834    1.073e-7, 9.918e-8, 9.191e-8, 8.538e-8, 7.949e-8, 7.419e-8,
01835    6.94e-8, 6.508e-8, 6.114e-8, 5.761e-8, 5.437e-8, 5.146e-8,
01836    4.89e-8, 4.636e-8, 4.406e-8, 4.201e-8, 4.015e-8, 3.84e-8,
01837    3.661e-8, 3.51e-8, 3.377e-8, 3.242e-8, 3.13e-8, 3.015e-8,
01838    2.918e-8, 2.83e-8, 2.758e-8, 2.707e-8, 2.656e-8, 2.619e-8,
01839    2.609e-8, 2.615e-8, 2.63e-8, 2.675e-8, 2.745e-8, 2.842e-8,
01840    2.966e-8, 3.125e-8, 3.318e-8, 3.565e-8, 3.85e-8, 4.191e-8,
01841    4.59e-8, 5.059e-8, 5.607e-8, 6.239e-8, 6.958e-8, 7.796e-8,
01842    8.773e-8, 9.88e-8, 1.114e-7, 1.258e-7, 1.422e-7, 1.61e-7,
01843    1.822e-7, 2.06e-7, 2.337e-7, 2.645e-7, 2.996e-7, 3.393e-7,
01844    3.843e-7, 4.363e-7, 4.935e-7, 5.607e-7, 6.363e-7, 7.242e-7,
01845    8.23e-7, 9.411e-7, 1.071e-6, 1.232e-6, 1.402e-6, 1.6e-6, 1.82e-6,
01846    2.128e-6, 2.386e-6, 2.781e-6, 3.242e-6, 3.653e-6, 4.323e-6,
01847    4.747e-6, 5.321e-6, 5.919e-6, 6.681e-6, 7.101e-6, 7.983e-6,
01848    8.342e-6, 8.741e-6, 9.431e-6, 9.952e-6, 1.026e-5, 1.055e-5,
01849    1.095e-5, 1.095e-5, 1.087e-5, 1.056e-5, 1.026e-5, 9.715e-6,
01850    9.252e-6, 8.452e-6, 7.958e-6, 7.268e-6, 6.295e-6, 6.003e-6, 5e-6,
01851    4.591e-6, 3.983e-6, 3.479e-6, 3.058e-6, 2.667e-6, 2.293e-6,
01852    1.995e-6, 1.747e-6, 1.517e-6, 1.335e-6, 1.165e-6, 1.028e-6,
01853    9.007e-7, 7.956e-7, 7.015e-7, 6.192e-7, 5.491e-7, 4.859e-7,
01854    4.297e-7, 3.799e-7, 3.38e-7, 3.002e-7, 2.659e-7, 2.366e-7,
01855    2.103e-7, 1.861e-7, 1.655e-7, 1.469e-7, 1.309e-7, 1.162e-7,
01856    1.032e-7, 9.198e-8, 8.181e-8, 7.294e-8, 6.516e-8, 5.787e-8,
01857    5.163e-8, 4.612e-8, 4.119e-8, 3.695e-8, 3.308e-8, 2.976e-8,
01858    2.67e-8, 2.407e-8, 2.171e-8, 1.965e-8, 1.78e-8, 1.617e-8,
01859    1.47e-8, 1.341e-8, 1.227e-8, 1.125e-8, 1.033e-8, 9.524e-9,
01860    8.797e-9, 8.162e-9, 7.565e-9, 7.04e-9, 6.56e-9, 6.129e-9,
01861    5.733e-9, 5.376e-9, 5.043e-9, 4.75e-9, 4.466e-9, 4.211e-9,
01862    3.977e-9, 3.759e-9, 3.558e-9, 3.373e-9, 3.201e-9, 3.043e-9,
01863    2.895e-9, 2.76e-9, 2.635e-9, 2.518e-9, 2.411e-9, 2.314e-9,
01864    2.23e-9, 2.151e-9, 2.087e-9, 2.035e-9, 1.988e-9, 1.946e-9,
01865    1.927e-9, 1.916e-9, 1.916e-9, 1.933e-9, 1.966e-9, 2.018e-9,
01866    2.09e-9, 2.182e-9, 2.299e-9, 2.442e-9, 2.623e-9, 2.832e-9,
01867    3.079e-9, 3.368e-9, 3.714e-9, 4.104e-9, 4.567e-9, 5.091e-9,
01868    5.701e-9, 6.398e-9, 7.194e-9, 8.127e-9, 9.141e-9, 1.035e-8,
01869    1.177e-8, 1.338e-8, 1.508e-8, 1.711e-8, 1.955e-8, 2.216e-8,
01870    2.534e-8, 2.871e-8, 3.291e-8, 3.711e-8, 4.285e-8, 4.868e-8,
01871    5.509e-8, 6.276e-8, 7.262e-8, 8.252e-8, 9.4e-8, 1.064e-7,
01872    1.247e-7, 1.411e-7, 1.626e-7, 1.827e-7, 2.044e-7, 2.284e-7,
01873    2.452e-7, 2.854e-7, 3.026e-7, 3.278e-7, 3.474e-7, 3.693e-7,
01874    3.93e-7, 4.104e-7, 4.22e-7, 4.439e-7, 4.545e-7, 4.778e-7,
01875    4.812e-7, 5.018e-7, 4.899e-7, 5.075e-7, 5.073e-7, 5.171e-7,
01876    5.131e-7, 5.25e-7, 5.617e-7, 5.846e-7, 6.239e-7, 6.696e-7,
01877    7.398e-7, 8.073e-7, 9.15e-7, 1.009e-6, 1.116e-6, 1.264e-6,
01878    1.439e-6, 1.644e-6, 1.856e-6, 2.147e-6, 2.317e-6, 2.713e-6,
01879    2.882e-6, 2.99e-6, 3.489e-6, 3.581e-6, 4.033e-6, 4.26e-6,
01880    4.543e-6, 4.84e-6, 4.826e-6, 5.013e-6, 5.252e-6, 5.277e-6,
01881    5.306e-6, 5.236e-6, 5.123e-6, 5.171e-6, 4.843e-6, 4.615e-6,
01882    4.385e-6, 3.97e-6, 3.693e-6, 3.231e-6, 2.915e-6, 2.495e-6,
01883    2.144e-6, 1.91e-6, 1.639e-6, 1.417e-6, 1.226e-6, 1.065e-6,
01884    9.29e-7, 8.142e-7, 7.161e-7, 6.318e-7, 5.581e-7, 4.943e-7,
01885    4.376e-7, 3.884e-7, 3.449e-7, 3.06e-7, 2.712e-7, 2.412e-7,
01886    2.139e-7, 1.903e-7, 1.689e-7, 1.499e-7, 1.331e-7, 1.183e-7,
01887    1.05e-7, 9.362e-8, 8.306e-8, 7.403e-8, 6.578e-8, 5.853e-8,
01888    5.216e-8, 4.632e-8, 4.127e-8, 3.678e-8, 3.279e-8, 2.923e-8,
01889    2.612e-8, 2.339e-8, 2.094e-8, 1.877e-8, 1.686e-8, 1.516e-8,

```

01890 1.366e-8, 1.234e-8, 1.114e-8, 1.012e-8, 9.182e-9, 8.362e-9,  
01891 7.634e-9, 6.981e-9, 6.406e-9, 5.888e-9, 5.428e-9, 5.021e-9,  
01892 4.65e-9, 4.326e-9, 4.033e-9, 3.77e-9, 3.536e-9, 3.327e-9,  
01893 3.141e-9, 2.974e-9, 2.825e-9, 2.697e-9, 2.584e-9, 2.488e-9,  
01894 2.406e-9, 2.34e-9, 2.292e-9, 2.259e-9, 2.244e-9, 2.243e-9,  
01895 2.272e-9, 2.31e-9, 2.378e-9, 2.454e-9, 2.618e-9, 2.672e-9,  
01896 2.831e-9, 3.05e-9, 3.225e-9, 3.425e-9, 3.677e-9, 3.968e-9,  
01897 4.221e-9, 4.639e-9, 4.96e-9, 5.359e-9, 5.649e-9, 6.23e-9,  
01898 6.716e-9, 7.218e-9, 7.746e-9, 7.988e-9, 8.627e-9, 8.999e-9,  
01899 9.442e-9, 9.82e-9, 1.015e-8, 1.06e-8, 1.079e-8, 1.109e-8,  
01900 1.137e-8, 1.186e-8, 1.18e-8, 1.187e-8, 1.194e-8, 1.192e-8,  
01901 1.224e-8, 1.245e-8, 1.246e-8, 1.318e-8, 1.377e-8, 1.471e-8,  
01902 1.582e-8, 1.713e-8, 1.853e-8, 2.063e-8, 2.27e-8, 2.567e-8,  
01903 2.891e-8, 3.264e-8, 3.744e-8, 4.286e-8, 4.915e-8, 5.623e-8,  
01904 6.336e-8, 7.293e-8, 8.309e-8, 9.319e-8, 1.091e-7, 1.243e-7,  
01905 1.348e-7, 1.449e-7, 1.62e-7, 1.846e-7, 1.937e-7, 2.04e-7,  
01906 2.179e-7, 2.298e-7, 2.433e-7, 2.439e-7, 2.464e-7, 2.611e-7,  
01907 2.617e-7, 2.582e-7, 2.453e-7, 2.401e-7, 2.349e-7, 2.203e-7,  
01908 2.066e-7, 1.939e-7, 1.78e-7, 1.558e-7, 1.391e-7, 1.203e-7,  
01909 1.048e-7, 9.464e-8, 8.306e-8, 7.239e-8, 6.317e-8, 5.52e-8,  
01910 4.847e-8, 4.282e-8, 3.796e-8, 3.377e-8, 2.996e-8, 2.678e-8,  
01911 2.4e-8, 2.134e-8, 1.904e-8, 1.705e-8, 1.523e-8, 1.35e-8,  
01912 1.204e-8, 1.07e-8, 9.408e-9, 8.476e-9, 7.47e-9, 6.679e-9,  
01913 5.929e-9, 5.267e-9, 4.711e-9, 4.172e-9, 3.761e-9, 3.288e-9,  
01914 2.929e-9, 2.609e-9, 2.315e-9, 2.042e-9, 1.844e-9, 1.64e-9,  
01915 1.47e-9, 1.31e-9, 1.176e-9, 1.049e-9, 9.377e-10, 8.462e-10,  
01916 7.616e-10, 6.854e-10, 6.191e-10, 5.596e-10, 5.078e-10, 4.611e-10,  
01917 4.197e-10, 3.83e-10, 3.505e-10, 3.215e-10, 2.956e-10, 2.726e-10,  
01918 2.521e-10, 2.338e-10, 2.173e-10, 2.026e-10, 1.895e-10, 1.777e-10,  
01919 1.672e-10, 1.579e-10, 1.496e-10, 1.423e-10, 1.358e-10, 1.302e-10,  
01920 1.254e-10, 1.216e-10, 1.187e-10, 1.163e-10, 1.147e-10, 1.145e-10,  
01921 1.15e-10, 1.17e-10, 1.192e-10, 1.25e-10, 1.298e-10, 1.345e-10,  
01922 1.405e-10, 1.538e-10, 1.648e-10, 1.721e-10, 1.872e-10, 1.968e-10,  
01923 2.089e-10, 2.172e-10, 2.317e-10, 2.389e-10, 2.503e-10, 2.585e-10,  
01924 2.686e-10, 2.8e-10, 2.895e-10, 3.019e-10, 3.037e-10, 3.076e-10,  
01925 3.146e-10, 3.198e-10, 3.332e-10, 3.397e-10, 3.54e-10, 3.667e-10,  
01926 3.895e-10, 4.071e-10, 4.565e-10, 4.983e-10, 5.439e-10, 5.968e-10,  
01927 6.676e-10, 7.456e-10, 8.405e-10, 9.478e-10, 1.064e-9, 1.218e-9,  
01928 1.386e-9, 1.581e-9, 1.787e-9, 2.032e-9, 2.347e-9, 2.677e-9,  
01929 3.008e-9, 3.544e-9, 4.056e-9, 4.687e-9, 5.331e-9, 6.227e-9,  
01930 6.854e-9, 8.139e-9, 8.945e-9, 9.865e-9, 1.125e-8, 1.178e-8,  
01931 1.364e-8, 1.436e-8, 1.54e-8, 1.672e-8, 1.793e-8, 1.906e-8,  
01932 2.036e-8, 2.144e-8, 2.292e-8, 2.371e-8, 2.493e-8, 2.606e-8,  
01933 2.706e-8, 2.866e-8, 3.036e-8, 3.136e-8, 3.405e-8, 3.665e-8,  
01934 3.837e-8, 4.229e-8, 4.748e-8, 5.32e-8, 5.763e-8, 6.677e-8,  
01935 7.216e-8, 7.716e-8, 8.958e-8, 9.419e-8, 1.036e-7, 1.108e-7,  
01936 1.189e-7, 1.246e-7, 1.348e-7, 1.31e-7, 1.361e-7, 1.364e-7,  
01937 1.363e-7, 1.343e-7, 1.293e-7, 1.254e-7, 1.235e-7, 1.158e-7,  
01938 1.107e-7, 9.961e-8, 9.011e-8, 7.91e-8, 6.916e-8, 6.338e-8,  
01939 5.564e-8, 4.827e-8, 4.198e-8, 3.695e-8, 3.276e-8, 2.929e-8,  
01940 2.633e-8, 2.391e-8, 2.192e-8, 2.021e-8, 1.89e-8, 1.772e-8,  
01941 1.667e-8, 1.603e-8, 1.547e-8, 1.537e-8, 1.492e-8, 1.515e-8,  
01942 1.479e-8, 1.45e-8, 1.513e-8, 1.495e-8, 1.529e-8, 1.565e-8,  
01943 1.564e-8, 1.553e-8, 1.569e-8, 1.584e-8, 1.57e-8, 1.538e-8,  
01944 1.513e-8, 1.472e-8, 1.425e-8, 1.349e-8, 1.328e-8, 1.249e-8,  
01945 1.17e-8, 1.077e-8, 9.514e-9, 8.614e-9, 7.46e-9, 6.621e-9,  
01946 5.775e-9, 5.006e-9, 4.308e-9, 3.747e-9, 3.24e-9, 2.84e-9,  
01947 2.481e-9, 2.184e-9, 1.923e-9, 1.71e-9, 1.504e-9, 1.334e-9,  
01948 1.187e-9, 1.053e-9, 9.367e-10, 8.306e-10, 7.419e-10, 6.63e-10,  
01949 5.918e-10, 5.277e-10, 4.717e-10, 4.222e-10, 3.783e-10, 3.39e-10,  
01950 3.036e-10, 2.729e-10, 2.455e-10, 2.211e-10, 1.995e-10, 1.804e-10,  
01951 1.635e-10, 1.485e-10, 1.355e-10, 1.24e-10, 1.139e-10, 1.051e-10,  
01952 9.757e-11, 9.114e-11, 8.577e-11, 8.139e-11, 7.792e-11, 7.52e-11,  
01953 7.39e-11, 7.311e-11, 7.277e-11, 7.482e-11, 7.698e-11, 8.162e-11,  
01954 8.517e-11, 8.968e-11, 9.905e-11, 1.075e-10, 1.187e-10, 1.291e-10,  
01955 1.426e-10, 1.573e-10, 1.734e-10, 1.905e-10, 2.097e-10, 2.28e-10,  
01956 2.473e-10, 2.718e-10, 2.922e-10, 3.128e-10, 3.361e-10, 3.641e-10,  
01957 3.91e-10, 4.196e-10, 4.501e-10, 4.932e-10, 5.258e-10, 5.755e-10,  
01958 6.253e-10, 6.664e-10, 7.344e-10, 7.985e-10, 8.877e-10, 1.005e-9,  
01959 1.118e-9, 1.251e-9, 1.428e-9, 1.61e-9, 1.888e-9, 2.077e-9,  
01960 2.331e-9, 2.751e-9, 3.061e-9, 3.522e-9, 3.805e-9, 4.181e-9,  
01961 4.575e-9, 5.167e-9, 5.634e-9, 6.007e-9, 6.501e-9, 6.829e-9,  
01962 7.211e-9, 7.262e-9, 7.696e-9, 7.832e-9, 7.799e-9, 7.651e-9,  
01963 7.304e-9, 7.15e-9, 6.977e-9, 6.603e-9, 6.209e-9, 5.69e-9,  
01964 5.432e-9, 4.764e-9, 4.189e-9, 3.64e-9, 3.203e-9, 2.848e-9,  
01965 2.51e-9, 2.194e-9, 1.946e-9, 1.75e-9, 1.567e-9, 1.426e-9,  
01966 1.302e-9, 1.197e-9, 1.109e-9, 1.035e-9, 9.719e-10, 9.207e-10,  
01967 8.957e-10, 8.578e-10, 8.262e-10, 8.117e-10, 7.987e-10, 7.875e-10,  
01968 7.741e-10, 7.762e-10, 7.537e-10, 7.424e-10, 7.474e-10, 7.294e-10,  
01969 7.216e-10, 7.233e-10, 7.075e-10, 6.892e-10, 6.618e-10, 6.314e-10,  
01970 6.208e-10, 5.689e-10, 5.55e-10, 4.984e-10, 4.6e-10, 4.078e-10,  
01971 3.879e-10, 3.459e-10, 2.982e-10, 2.626e-10, 2.329e-10, 1.988e-10,  
01972 1.735e-10, 1.487e-10, 1.297e-10, 1.133e-10, 9.943e-11, 8.736e-11,  
01973 7.726e-11, 6.836e-11, 6.053e-11, 5.384e-11, 4.789e-11, 4.267e-11,  
01974 3.804e-11, 3.398e-11, 3.034e-11, 2.71e-11, 2.425e-11, 2.173e-11,  
01975 1.95e-11, 1.752e-11, 1.574e-11, 1.418e-11, 1.278e-11, 1.154e-11,  
01976 1.044e-11, 9.463e-12, 8.602e-12, 7.841e-12, 7.171e-12, 6.584e-12,

```
01977 6.073e-12, 5.631e-12, 5.254e-12, 4.937e-12, 4.679e-12, 4.476e-12,
01978 4.328e-12, 4.233e-12, 4.194e-12, 4.211e-12, 4.286e-12, 4.424e-12,
01979 4.628e-12, 4.906e-12, 5.262e-12, 5.708e-12, 6.254e-12, 6.914e-12,
01980 7.714e-12, 8.677e-12, 9.747e-12, 1.101e-11, 1.256e-11, 1.409e-11,
01981 1.597e-11, 1.807e-11, 2.034e-11, 2.316e-11, 2.622e-11, 2.962e-11,
01982 3.369e-11, 3.819e-11, 4.329e-11, 4.932e-11, 5.589e-11, 6.364e-11,
01983 7.284e-11, 8.236e-11, 9.447e-11, 1.078e-10, 1.229e-10, 1.417e-10,
01984 1.614e-10, 1.843e-10, 2.107e-10, 2.406e-10, 2.728e-10, 3.195e-10,
01985 3.595e-10, 4.153e-10, 4.736e-10, 5.41e-10, 6.088e-10, 6.769e-10,
01986 7.691e-10, 8.545e-10, 9.621e-10, 1.047e-9, 1.161e-9, 1.296e-9,
01987 1.424e-9, 1.576e-9, 1.739e-9, 1.893e-9, 2.08e-9, 2.336e-9,
01988 2.604e-9, 2.76e-9, 3.001e-9, 3.365e-9, 3.55e-9, 3.895e-9,
01989 4.183e-9, 4.614e-9, 4.846e-9, 5.068e-9, 5.427e-9, 5.541e-9,
01990 5.864e-9, 5.997e-9, 5.997e-9, 6.061e-9, 5.944e-9, 5.855e-9,
01991 5.661e-9, 5.523e-9, 5.374e-9, 4.94e-9, 4.688e-9, 4.17e-9,
01992 3.913e-9, 3.423e-9, 2.997e-9, 2.598e-9, 2.253e-9, 1.946e-9,
01993 1.71e-9, 1.507e-9, 1.336e-9, 1.19e-9, 1.068e-9, 9.623e-10,
01994 8.772e-10, 8.007e-10, 7.42e-10, 6.884e-10, 6.483e-10, 6.162e-10,
01995 5.922e-10, 5.688e-10, 5.654e-10, 5.637e-10, 5.701e-10, 5.781e-10,
01996 5.874e-10, 6.268e-10, 6.357e-10, 6.525e-10, 7.137e-10, 7.441e-10,
01997 8.024e-10, 8.485e-10, 9.143e-10, 9.536e-10, 9.717e-10, 1.018e-9,
01998 1.042e-9, 1.054e-9, 1.092e-9, 1.079e-9, 1.064e-9, 1.043e-9,
01999 1.02e-9, 9.687e-10, 9.273e-10, 9.208e-10, 9.068e-10, 7.687e-10,
02000 7.385e-10, 6.595e-10, 5.87e-10, 5.144e-10, 4.417e-10, 3.804e-10,
02001 3.301e-10, 2.866e-10, 2.509e-10, 2.202e-10, 1.947e-10, 1.719e-10,
02002 1.525e-10, 1.361e-10, 1.21e-10, 1.084e-10, 9.8e-11, 8.801e-11,
02003 7.954e-11, 7.124e-11, 6.335e-11, 5.76e-11, 5.132e-11, 4.601e-11,
02004 4.096e-11, 3.657e-11, 3.25e-11, 2.909e-11, 2.587e-11, 2.297e-11,
02005 2.05e-11, 1.828e-11, 1.632e-11, 1.462e-11, 1.314e-11, 1.185e-11,
02006 1.073e-11, 9.76e-12, 8.922e-12, 8.206e-12, 7.602e-12, 7.1e-12,
02007 6.694e-12, 6.378e-12, 6.149e-12, 6.004e-12, 5.941e-12, 5.962e-12,
02008 6.069e-12, 6.265e-12, 6.551e-12, 6.935e-12, 7.457e-12, 8.074e-12,
02009 8.811e-12, 9.852e-12, 1.086e-11, 1.207e-11, 1.361e-11, 1.553e-11,
02010 1.737e-11, 1.93e-11, 2.175e-11, 2.41e-11, 2.706e-11, 3.023e-11,
02011 3.313e-11, 3.657e-11, 4.118e-11, 4.569e-11, 5.025e-11, 5.66e-11,
02012 6.231e-11, 6.881e-11, 7.996e-11, 8.526e-11, 9.694e-11, 1.106e-10,
02013 1.222e-10, 1.355e-10, 1.525e-10, 1.775e-10, 1.924e-10, 2.181e-10,
02014 2.379e-10, 2.662e-10, 2.907e-10, 3.154e-10, 3.366e-10, 3.579e-10,
02015 3.858e-10, 4.046e-10, 4.196e-10, 4.166e-10, 4.457e-10, 4.466e-10,
02016 4.404e-10, 4.337e-10, 4.15e-10, 4.083e-10, 3.91e-10, 3.723e-10,
02017 3.514e-10, 3.303e-10, 2.847e-10, 2.546e-10, 2.23e-10, 1.994e-10,
02018 1.733e-10, 1.488e-10, 1.297e-10, 1.144e-10, 1.004e-10, 8.741e-11,
02019 7.928e-11, 7.034e-11, 6.323e-11, 5.754e-11, 5.25e-11, 4.85e-11,
02020 4.502e-11, 4.286e-11, 4.028e-11, 3.899e-11, 3.824e-11, 3.761e-11,
02021 3.804e-11, 3.839e-11, 3.845e-11, 4.244e-11, 4.382e-11, 4.582e-11,
02022 4.847e-11, 5.209e-11, 5.384e-11, 5.887e-11, 6.371e-11, 6.737e-11,
02023 7.168e-11, 7.415e-11, 7.827e-11, 8.037e-11, 8.12e-11, 8.071e-11,
02024 8.008e-11, 7.851e-11, 7.544e-11, 7.377e-11, 7.173e-11, 6.801e-11,
02025 6.267e-11, 5.727e-11, 5.288e-11, 4.853e-11, 4.082e-11, 3.645e-11,
02026 3.136e-11, 2.672e-11, 2.304e-11, 1.986e-11, 1.725e-11, 1.503e-11,
02027 1.315e-11, 1.153e-11, 1.014e-11, 8.942e-12, 7.901e-12, 6.993e-12,
02028 6.199e-12, 5.502e-12, 4.89e-12, 4.351e-12, 3.878e-12, 3.461e-12,
02029 3.094e-12, 2.771e-12, 2.488e-12, 2.241e-12, 2.025e-12, 1.838e-12,
02030 1.677e-12, 1.541e-12, 1.427e-12, 1.335e-12, 1.262e-12, 1.209e-12,
02031 1.176e-12, 1.161e-12, 1.165e-12, 1.189e-12, 1.234e-12, 1.3e-12,
02032 1.389e-12, 1.503e-12, 1.644e-12, 1.814e-12, 2.017e-12, 2.255e-12,
02033 2.534e-12, 2.858e-12, 3.231e-12, 3.661e-12, 4.153e-12, 4.717e-12,
02034 5.36e-12, 6.094e-12, 6.93e-12, 7.882e-12, 8.966e-12, 1.02e-11,
02035 1.162e-11, 1.324e-11, 1.51e-11, 1.72e-11, 1.965e-11, 2.237e-11,
02036 2.56e-11, 2.927e-11, 3.371e-11, 3.842e-11, 4.429e-11, 5.139e-11,
02037 5.798e-11, 6.697e-11, 7.626e-11, 8.647e-11, 1.022e-10, 1.136e-10,
02038 1.3e-10, 1.481e-10, 1.672e-10, 1.871e-10, 2.126e-10, 2.357e-10,
02039 2.583e-10, 2.997e-10, 3.289e-10, 3.702e-10, 4.012e-10, 4.319e-10,
02040 4.527e-10, 5.001e-10, 5.448e-10, 5.611e-10, 5.76e-10, 5.965e-10,
02041 6.079e-10, 6.207e-10, 6.276e-10, 6.222e-10, 6.137e-10, 6e-10,
02042 5.814e-10, 5.393e-10, 5.35e-10, 4.947e-10, 4.629e-10, 4.117e-10,
02043 3.712e-10, 3.372e-10, 2.923e-10, 2.55e-10, 2.232e-10, 1.929e-10,
02044 1.679e-10, 1.46e-10, 1.289e-10, 1.13e-10, 9.953e-11, 8.763e-11,
02045 7.76e-11, 6.9e-11, 6.16e-11, 5.525e-11, 4.958e-11, 4.489e-11,
02046 4.072e-11, 3.728e-11, 3.438e-11, 3.205e-11, 3.006e-11, 2.848e-11,
02047 2.766e-11, 2.688e-11, 2.664e-11, 2.67e-11, 2.696e-11, 2.786e-11,
02048 2.861e-11, 3.009e-11, 3.178e-11, 3.389e-11, 3.587e-11, 3.819e-11,
02049 4.054e-11, 4.417e-11, 4.703e-11, 5.137e-11, 5.46e-11, 6.055e-11,
02050 6.333e-11, 6.773e-11, 7.219e-11, 7.717e-11, 8.131e-11, 8.491e-11,
02051 8.574e-11, 9.01e-11, 9.017e-11, 8.999e-11, 8.959e-11, 8.838e-11,
02052 8.579e-11, 8.162e-11, 8.098e-11, 7.472e-11, 7.108e-11, 6.559e-11,
02053 5.994e-11, 5.172e-11, 4.424e-11, 3.951e-11, 3.34e-11, 2.902e-11,
02054 2.541e-11, 2.215e-11, 1.945e-11, 1.716e-11, 1.503e-11, 1.339e-11,
02055 1.185e-11, 1.05e-11, 9.336e-12, 8.307e-12, 7.312e-12, 6.55e-12,
02056 5.836e-12, 5.178e-12, 4.6e-12, 4.086e-12, 3.639e-12, 3.247e-12,
02057 2.904e-12, 2.604e-12, 2.341e-12, 2.112e-12, 1.914e-12, 1.744e-12,
02058 1.598e-12, 1.476e-12, 1.374e-12, 1.293e-12, 1.23e-12, 1.185e-12,
02059 1.158e-12, 1.147e-12, 1.154e-12, 1.177e-12, 1.219e-12, 1.28e-12,
02060 1.36e-12, 1.463e-12, 1.591e-12, 1.75e-12, 1.94e-12, 2.156e-12,
02061 2.43e-12, 2.748e-12, 3.052e-12, 3.533e-12, 3.967e-12, 4.471e-12,
02062 5.041e-12, 5.86e-12, 6.664e-12, 7.522e-12, 8.342e-12, 9.412e-12,
02063 1.072e-11, 1.213e-11, 1.343e-11, 1.496e-11, 1.664e-11, 1.822e-11,
```

```
02064 2.029e-11, 2.233e-11, 2.457e-11, 2.709e-11, 2.928e-11, 3.115e-11,
02065 3.356e-11, 3.592e-11, 3.818e-11, 3.936e-11, 4.061e-11, 4.149e-11,
02066 4.299e-11, 4.223e-11, 4.251e-11, 4.287e-11, 4.177e-11, 4.094e-11,
02067 3.942e-11, 3.772e-11, 3.614e-11, 3.394e-11, 3.222e-11, 2.791e-11,
02068 2.665e-11, 2.309e-11, 2.032e-11, 1.74e-11, 1.535e-11, 1.323e-11,
02069 1.151e-11, 9.803e-12, 8.65e-12, 7.54e-12, 6.619e-12, 5.832e-12,
02070 5.113e-12, 4.503e-12, 3.975e-12, 3.52e-12, 3.112e-12, 2.797e-12,
02071 2.5e-12, 2.24e-12, 2.013e-12, 1.819e-12, 1.653e-12, 1.513e-12,
02072 1.395e-12, 1.299e-12, 1.225e-12, 1.168e-12, 1.124e-12, 1.148e-12,
02073 1.107e-12, 1.128e-12, 1.169e-12, 1.233e-12, 1.307e-12, 1.359e-12,
02074 1.543e-12, 1.686e-12, 1.794e-12, 2.028e-12, 2.21e-12, 2.441e-12,
02075 2.653e-12, 2.828e-12, 3.093e-12, 3.28e-12, 3.551e-12, 3.677e-12,
02076 3.803e-12, 3.844e-12, 4.068e-12, 4.093e-12, 4.002e-12, 3.904e-12,
02077 3.624e-12, 3.633e-12, 3.622e-12, 3.443e-12, 3.184e-12, 2.934e-12,
02078 2.476e-12, 2.212e-12, 1.867e-12, 1.594e-12, 1.37e-12, 1.192e-12,
02079 1.045e-12, 9.211e-13, 8.17e-13, 7.29e-13, 6.55e-13, 5.929e-13,
02080 5.415e-13, 4.995e-13, 4.661e-13, 4.406e-13, 4.225e-13, 4.116e-13,
02081 4.075e-13, 4.102e-13, 4.198e-13, 4.365e-13, 4.606e-13, 4.925e-13,
02082 5.326e-13, 5.818e-13, 6.407e-13, 7.104e-13, 7.92e-13, 8.868e-13,
02083 9.964e-13, 1.123e-12, 1.268e-12, 1.434e-12, 1.626e-12, 1.848e-12,
02084 2.107e-12, 2.422e-12, 2.772e-12, 3.145e-12, 3.704e-12, 4.27e-12,
02085 4.721e-12, 5.361e-12, 6.083e-12, 7.095e-12, 7.968e-12, 9.228e-12,
02086 1.048e-11, 1.187e-11, 1.336e-11, 1.577e-11, 1.772e-11, 2.017e-11,
02087 2.25e-11, 2.63e-11, 2.911e-11, 3.356e-11, 3.82e-11, 4.173e-11,
02088 4.811e-11, 5.254e-11, 5.839e-11, 6.187e-11, 6.805e-11, 7.118e-11,
02089 7.369e-11, 7.664e-11, 7.794e-11, 7.947e-11, 8.036e-11, 7.954e-11,
02090 7.849e-11, 7.518e-11, 7.462e-11, 6.926e-11, 6.531e-11, 6.197e-11,
02091 5.421e-11, 4.777e-11, 4.111e-11, 3.679e-11, 3.166e-11, 2.786e-11,
02092 2.436e-11, 2.144e-11, 1.859e-11, 1.628e-11, 1.414e-11, 1.237e-11,
02093 1.093e-11, 9.558e-12
02094 };
02095
02096 static double h2o260[2001] = { .2752, .2732, .2749, .2676, .2667, .2545,
02097 .2497, .2327, .2218, .2036, .1825, .1694, .1497, .1353, .121,
02098 .1014, .09405, .07848, .07195, .06246, .05306, .04853, .04138,
02099 .03735, .03171, .02785, .02431, .02111, .01845, .0164, .01405,
02100 .01255, .01098, .009797, .008646, .007779, .006898, .006099,
02101 .005453, .004909, .004413, .003959, .003581, .003199, .002871,
02102 .002583, .00233, .002086, .001874, .001684, .001512, .001361,
02103 .001225, .0011, 9.89e-4, 8.916e-4, 8.039e-4, 7.256e-4, 6.545e-4,
02104 5.918e-4, 5.359e-4, 4.867e-4, 4.426e-4, 4.033e-4, 3.682e-4,
02105 3.366e-4, 3.088e-4, 2.833e-4, 2.605e-4, 2.403e-4, 2.221e-4,
02106 2.055e-4, 1.908e-4, 1.774e-4, 1.653e-4, 1.544e-4, 1.443e-4,
02107 1.351e-4, 1.267e-4, 1.19e-4, 1.119e-4, 1.053e-4, 9.922e-5,
02108 9.355e-5, 8.831e-5, 8.339e-5, 7.878e-5, 7.449e-5, 7.043e-5,
02109 6.664e-5, 6.307e-5, 5.969e-5, 5.654e-5, 5.357e-5, 5.075e-5,
02110 4.81e-5, 4.56e-5, 4.322e-5, 4.102e-5, 3.892e-5, 3.696e-5,
02111 3.511e-5, 3.339e-5, 3.177e-5, 3.026e-5, 2.886e-5, 2.756e-5,
02112 2.636e-5, 2.527e-5, 2.427e-5, 2.337e-5, 2.257e-5, 2.185e-5,
02113 2.127e-5, 2.08e-5, 2.041e-5, 2.013e-5, 2e-5, 1.997e-5, 2.009e-5,
02114 2.031e-5, 2.068e-5, 2.124e-5, 2.189e-5, 2.267e-5, 2.364e-5,
02115 2.463e-5, 2.618e-5, 2.774e-5, 2.937e-5, 3.144e-5, 3.359e-5,
02116 3.695e-5, 4.002e-5, 4.374e-5, 4.947e-5, 5.431e-5, 6.281e-5,
02117 7.169e-5, 8.157e-5, 9.728e-5, 1.079e-4, 1.337e-4, 1.442e-4,
02118 1.683e-4, 1.879e-4, 2.223e-4, 2.425e-4, 2.838e-4, 3.143e-4,
02119 3.527e-4, 4.012e-4, 4.237e-4, 4.747e-4, 5.057e-4, 5.409e-4,
02120 5.734e-4, 5.944e-4, 6.077e-4, 6.175e-4, 6.238e-4, 6.226e-4,
02121 6.248e-4, 6.192e-4, 6.098e-4, 5.818e-4, 5.709e-4, 5.465e-4,
02122 5.043e-4, 4.699e-4, 4.294e-4, 3.984e-4, 3.672e-4, 3.152e-4,
02123 2.883e-4, 2.503e-4, 2.211e-4, 1.92e-4, 1.714e-4, 1.485e-4,
02124 1.358e-4, 1.156e-4, 1.021e-4, 8.887e-5, 7.842e-5, 7.12e-5,
02125 6.186e-5, 5.73e-5, 4.792e-5, 4.364e-5, 3.72e-5, 3.28e-5,
02126 2.946e-5, 2.591e-5, 2.261e-5, 2.048e-5, 1.813e-5, 1.63e-5,
02127 1.447e-5, 1.282e-5, 1.167e-5, 1.041e-5, 9.449e-6, 8.51e-6,
02128 7.596e-6, 6.961e-6, 6.272e-6, 5.728e-6, 5.198e-6, 4.667e-6,
02129 4.288e-6, 3.897e-6, 3.551e-6, 3.235e-6, 2.952e-6, 2.688e-6,
02130 2.449e-6, 2.241e-6, 2.05e-6, 1.879e-6, 1.722e-6, 1.582e-6,
02131 1.456e-6, 1.339e-6, 1.236e-6, 1.144e-6, 1.06e-6, 9.83e-7,
02132 9.149e-7, 8.535e-7, 7.973e-7, 7.466e-7, 6.999e-7, 6.574e-7,
02133 6.18e-7, 5.821e-7, 5.487e-7, 5.18e-7, 4.896e-7, 4.631e-7,
02134 4.386e-7, 4.16e-7, 3.945e-7, 3.748e-7, 3.562e-7, 3.385e-7,
02135 3.222e-7, 3.068e-7, 2.922e-7, 2.788e-7, 2.659e-7, 2.539e-7,
02136 2.425e-7, 2.318e-7, 2.219e-7, 2.127e-7, 2.039e-7, 1.958e-7,
02137 1.885e-7, 1.818e-7, 1.758e-7, 1.711e-7, 1.662e-7, 1.63e-7,
02138 1.605e-7, 1.58e-7, 1.559e-7, 1.545e-7, 1.532e-7, 1.522e-7,
02139 1.51e-7, 1.495e-7, 1.465e-7, 1.483e-7, 1.469e-7, 1.448e-7,
02140 1.444e-7, 1.436e-7, 1.426e-7, 1.431e-7, 1.425e-7, 1.445e-7,
02141 1.477e-7, 1.515e-7, 1.567e-7, 1.634e-7, 1.712e-7, 1.802e-7,
02142 1.914e-7, 2.024e-7, 2.159e-7, 2.295e-7, 2.461e-7, 2.621e-7,
02143 2.868e-7, 3.102e-7, 3.394e-7, 3.784e-7, 4.223e-7, 4.864e-7,
02144 5.501e-7, 6.039e-7, 7.193e-7, 7.728e-7, 9.514e-7, 1.073e-6,
02145 1.18e-6, 1.333e-6, 1.472e-6, 1.566e-6, 1.677e-6, 1.784e-6,
02146 1.904e-6, 1.953e-6, 2.02e-6, 2.074e-6, 2.128e-6, 2.162e-6,
02147 2.219e-6, 2.221e-6, 2.249e-6, 2.239e-6, 2.235e-6, 2.185e-6,
02148 2.141e-6, 2.124e-6, 2.09e-6, 2.068e-6, 2.1e-6, 2.104e-6,
02149 2.142e-6, 2.181e-6, 2.257e-6, 2.362e-6, 2.5e-6, 2.664e-6,
02150 2.884e-6, 3.189e-6, 3.48e-6, 3.847e-6, 4.313e-6, 4.79e-6,
```

```
02151 5.25e-6, 5.989e-6, 6.692e-6, 7.668e-6, 8.52e-6, 9.606e-6,
02152 1.073e-5, 1.225e-5, 1.377e-5, 1.582e-5, 1.761e-5, 2.029e-5,
02153 2.284e-5, 2.602e-5, 2.94e-5, 3.483e-5, 3.928e-5, 4.618e-5,
02154 5.24e-5, 6.132e-5, 7.183e-5, 8.521e-5, 9.111e-5, 1.07e-4,
02155 1.184e-4, 1.264e-4, 1.475e-4, 1.612e-4, 1.704e-4, 1.818e-4,
02156 1.924e-4, 1.994e-4, 2.061e-4, 2.18e-4, 2.187e-4, 2.2e-4,
02157 2.196e-4, 2.131e-4, 2.015e-4, 1.988e-4, 1.847e-4, 1.729e-4,
02158 1.597e-4, 1.373e-4, 1.262e-4, 1.087e-4, 9.439e-5, 8.061e-5,
02159 7.093e-5, 6.049e-5, 5.12e-5, 4.435e-5, 3.817e-5, 3.34e-5,
02160 2.927e-5, 2.573e-5, 2.291e-5, 2.04e-5, 1.827e-5, 1.636e-5,
02161 1.463e-5, 1.309e-5, 1.17e-5, 1.047e-5, 9.315e-6, 8.328e-6,
02162 7.458e-6, 6.665e-6, 5.94e-6, 5.316e-6, 4.752e-6, 4.252e-6,
02163 3.825e-6, 3.421e-6, 3.064e-6, 2.746e-6, 2.465e-6, 2.216e-6,
02164 1.99e-6, 1.79e-6, 1.609e-6, 1.449e-6, 1.306e-6, 1.177e-6,
02165 1.063e-6, 9.607e-7, 8.672e-7, 7.855e-7, 7.118e-7, 6.46e-7,
02166 5.871e-7, 5.34e-7, 4.868e-7, 4.447e-7, 4.068e-7, 3.729e-7,
02167 3.423e-7, 3.151e-7, 2.905e-7, 2.686e-7, 2.484e-7, 2.306e-7,
02168 2.142e-7, 1.995e-7, 1.86e-7, 1.738e-7, 1.626e-7, 1.522e-7,
02169 1.427e-7, 1.338e-7, 1.258e-7, 1.183e-7, 1.116e-7, 1.056e-7,
02170 9.972e-8, 9.46e-8, 9.007e-8, 8.592e-8, 8.195e-8, 7.816e-8,
02171 7.483e-8, 7.193e-8, 6.892e-8, 6.642e-8, 6.386e-8, 6.154e-8,
02172 5.949e-8, 5.764e-8, 5.622e-8, 5.479e-8, 5.364e-8, 5.301e-8,
02173 5.267e-8, 5.263e-8, 5.313e-8, 5.41e-8, 5.55e-8, 5.745e-8,
02174 6.003e-8, 6.311e-8, 6.713e-8, 7.173e-8, 7.724e-8, 8.368e-8,
02175 9.121e-8, 9.986e-8, 1.097e-7, 1.209e-7, 1.338e-7, 1.486e-7,
02176 1.651e-7, 1.837e-7, 2.048e-7, 2.289e-7, 2.557e-7, 2.857e-7,
02177 3.195e-7, 3.587e-7, 4.015e-7, 4.497e-7, 5.049e-7, 5.665e-7,
02178 6.366e-7, 7.121e-7, 7.996e-7, 8.946e-7, 1.002e-6, 1.117e-6,
02179 1.262e-6, 1.416e-6, 1.611e-6, 1.807e-6, 2.056e-6, 2.351e-6,
02180 2.769e-6, 3.138e-6, 3.699e-6, 4.386e-6, 5.041e-6, 6.074e-6,
02181 6.812e-6, 7.79e-6, 8.855e-6, 1.014e-5, 1.095e-5, 1.245e-5,
02182 1.316e-5, 1.39e-5, 1.504e-5, 1.583e-5, 1.617e-5, 1.652e-5,
02183 1.713e-5, 1.724e-5, 1.715e-5, 1.668e-5, 1.629e-5, 1.552e-5,
02184 1.478e-5, 1.34e-5, 1.245e-5, 1.121e-5, 9.575e-6, 8.956e-6,
02185 7.345e-6, 6.597e-6, 5.612e-6, 4.818e-6, 4.165e-6, 3.579e-6,
02186 3.041e-6, 2.623e-6, 2.29e-6, 1.984e-6, 1.748e-6, 1.534e-6,
02187 1.369e-6, 1.219e-6, 1.092e-6, 9.8e-7, 8.762e-7, 7.896e-7,
02188 7.104e-7, 6.364e-7, 5.691e-7, 5.107e-7, 4.575e-7, 4.09e-7,
02189 3.667e-7, 3.287e-7, 2.931e-7, 2.633e-7, 2.356e-7, 2.111e-7,
02190 1.895e-7, 1.697e-7, 1.525e-7, 1.369e-7, 1.233e-7, 1.114e-7,
02191 9.988e-8, 9.004e-8, 8.149e-8, 7.352e-8, 6.662e-8, 6.03e-8,
02192 5.479e-8, 4.974e-8, 4.532e-8, 4.129e-8, 3.781e-8, 3.462e-8,
02193 3.176e-8, 2.919e-8, 2.687e-8, 2.481e-8, 2.292e-8, 2.119e-8,
02194 1.967e-8, 1.828e-8, 1.706e-8, 1.589e-8, 1.487e-8, 1.393e-8,
02195 1.307e-8, 1.228e-8, 1.156e-8, 1.089e-8, 1.028e-8, 9.696e-9,
02196 9.159e-9, 8.658e-9, 8.187e-9, 7.746e-9, 7.34e-9, 6.953e-9,
02197 6.594e-9, 6.259e-9, 5.948e-9, 5.66e-9, 5.386e-9, 5.135e-9,
02198 4.903e-9, 4.703e-9, 4.515e-9, 4.362e-9, 4.233e-9, 4.117e-9,
02199 4.017e-9, 3.962e-9, 3.924e-9, 3.905e-9, 3.922e-9, 3.967e-9,
02200 4.046e-9, 4.165e-9, 4.32e-9, 4.522e-9, 4.769e-9, 5.083e-9,
02201 5.443e-9, 5.872e-9, 6.366e-9, 6.949e-9, 7.601e-9, 8.371e-9,
02202 9.22e-9, 1.02e-8, 1.129e-8, 1.251e-8, 1.393e-8, 1.542e-8,
02203 1.72e-8, 1.926e-8, 2.152e-8, 2.392e-8, 2.678e-8, 3.028e-8,
02204 3.39e-8, 3.836e-8, 4.309e-8, 4.9e-8, 5.481e-8, 6.252e-8,
02205 7.039e-8, 7.883e-8, 8.849e-8, 1.012e-7, 1.142e-7, 1.3e-7,
02206 1.475e-7, 1.732e-7, 1.978e-7, 2.304e-7, 2.631e-7, 2.988e-7,
02207 3.392e-7, 3.69e-7, 4.355e-7, 4.672e-7, 5.11e-7, 5.461e-7,
02208 5.828e-7, 6.233e-7, 6.509e-7, 6.672e-7, 6.969e-7, 7.104e-7,
02209 7.439e-7, 7.463e-7, 7.708e-7, 7.466e-7, 7.668e-7, 7.549e-7,
02210 7.586e-7, 7.384e-7, 7.439e-7, 7.785e-7, 7.915e-7, 8.31e-7,
02211 8.745e-7, 9.558e-7, 1.038e-6, 1.173e-6, 1.304e-6, 1.452e-6,
02212 1.671e-6, 1.931e-6, 2.239e-6, 2.578e-6, 3.032e-6, 3.334e-6,
02213 3.98e-6, 4.3e-6, 4.518e-6, 5.321e-6, 5.508e-6, 6.211e-6, 6.59e-6,
02214 7.046e-6, 7.555e-6, 7.558e-6, 7.875e-6, 8.319e-6, 8.433e-6,
02215 8.59e-6, 8.503e-6, 8.304e-6, 8.336e-6, 7.739e-6, 7.301e-6,
02216 6.827e-6, 6.078e-6, 5.551e-6, 4.762e-6, 4.224e-6, 3.538e-6,
02217 2.984e-6, 2.619e-6, 2.227e-6, 1.923e-6, 1.669e-6, 1.462e-6,
02218 1.294e-6, 1.155e-6, 1.033e-6, 9.231e-7, 8.238e-7, 7.36e-7,
02219 6.564e-7, 5.869e-7, 5.236e-7, 4.673e-7, 4.174e-7, 3.736e-7,
02220 3.33e-7, 2.976e-7, 2.657e-7, 2.367e-7, 2.106e-7, 1.877e-7,
02221 1.671e-7, 1.494e-7, 1.332e-7, 1.192e-7, 1.065e-7, 9.558e-8,
02222 8.586e-8, 7.717e-8, 6.958e-8, 6.278e-8, 5.666e-8, 5.121e-8,
02223 4.647e-8, 4.213e-8, 3.815e-8, 3.459e-8, 3.146e-8, 2.862e-8,
02224 2.604e-8, 2.375e-8, 2.162e-8, 1.981e-8, 1.817e-8, 1.67e-8,
02225 1.537e-8, 1.417e-8, 1.31e-8, 1.215e-8, 1.128e-8, 1.05e-8,
02226 9.793e-9, 9.158e-9, 8.586e-9, 8.068e-9, 7.595e-9, 7.166e-9,
02227 6.778e-9, 6.427e-9, 6.108e-9, 5.826e-9, 5.571e-9, 5.347e-9,
02228 5.144e-9, 4.968e-9, 4.822e-9, 4.692e-9, 4.589e-9, 4.506e-9,
02229 4.467e-9, 4.44e-9, 4.466e-9, 4.515e-9, 4.718e-9, 4.729e-9,
02230 4.937e-9, 5.249e-9, 5.466e-9, 5.713e-9, 6.03e-9, 6.436e-9,
02231 6.741e-9, 7.33e-9, 7.787e-9, 8.414e-9, 8.908e-9, 9.868e-9,
02232 1.069e-8, 1.158e-8, 1.253e-8, 1.3e-8, 1.409e-8, 1.47e-8,
02233 1.548e-8, 1.612e-8, 1.666e-8, 1.736e-8, 1.763e-8, 1.812e-8,
02234 1.852e-8, 1.923e-8, 1.897e-8, 1.893e-8, 1.888e-8, 1.868e-8,
02235 1.895e-8, 1.899e-8, 1.876e-8, 1.96e-8, 2.02e-8, 2.121e-8,
02236 2.239e-8, 2.379e-8, 2.526e-8, 2.766e-8, 2.994e-8, 3.332e-8,
02237 3.703e-8, 4.158e-8, 4.774e-8, 5.499e-8, 6.355e-8, 7.349e-8,
```

02238 8.414e-8, 9.846e-8, 1.143e-7, 1.307e-7, 1.562e-7, 1.817e-7,  
02239 2.011e-7, 2.192e-7, 2.485e-7, 2.867e-7, 3.035e-7, 3.223e-7,  
02240 3.443e-7, 3.617e-7, 3.793e-7, 3.793e-7, 3.839e-7, 4.081e-7,  
02241 4.117e-7, 4.085e-7, 3.92e-7, 3.851e-7, 3.754e-7, 3.49e-7,  
02242 3.229e-7, 2.978e-7, 2.691e-7, 2.312e-7, 2.029e-7, 1.721e-7,  
02243 1.472e-7, 1.308e-7, 1.132e-7, 9.736e-8, 8.458e-8, 7.402e-8,  
02244 6.534e-8, 5.811e-8, 5.235e-8, 4.762e-8, 4.293e-8, 3.896e-8,  
02245 3.526e-8, 3.165e-8, 2.833e-8, 2.551e-8, 2.288e-8, 2.036e-8,  
02246 1.82e-8, 1.626e-8, 1.438e-8, 1.299e-8, 1.149e-8, 1.03e-8,  
02247 9.148e-9, 8.122e-9, 7.264e-9, 6.425e-9, 5.777e-9, 5.06e-9,  
02248 4.502e-9, 4.013e-9, 3.567e-9, 3.145e-9, 2.864e-9, 2.553e-9,  
02249 2.311e-9, 2.087e-9, 1.886e-9, 1.716e-9, 1.556e-9, 1.432e-9,  
02250 1.311e-9, 1.202e-9, 1.104e-9, 1.013e-9, 9.293e-10, 8.493e-10,  
02251 7.79e-10, 7.185e-10, 6.642e-10, 6.141e-10, 5.684e-10, 5.346e-10,  
02252 5.032e-10, 4.725e-10, 4.439e-10, 4.176e-10, 3.93e-10, 3.714e-10,  
02253 3.515e-10, 3.332e-10, 3.167e-10, 3.02e-10, 2.887e-10, 2.769e-10,  
02254 2.665e-10, 2.578e-10, 2.503e-10, 2.436e-10, 2.377e-10, 2.342e-10,  
02255 2.305e-10, 2.296e-10, 2.278e-10, 2.321e-10, 2.355e-10, 2.402e-10,  
02256 2.478e-10, 2.67e-10, 2.848e-10, 2.982e-10, 3.263e-10, 3.438e-10,  
02257 3.649e-10, 3.829e-10, 4.115e-10, 4.264e-10, 4.473e-10, 4.63e-10,  
02258 4.808e-10, 4.995e-10, 5.142e-10, 5.313e-10, 5.318e-10, 5.358e-10,  
02259 5.452e-10, 5.507e-10, 5.698e-10, 5.782e-10, 5.983e-10, 6.164e-10,  
02260 6.532e-10, 6.811e-10, 7.624e-10, 8.302e-10, 9.067e-10, 9.937e-10,  
02261 1.104e-9, 1.221e-9, 1.361e-9, 1.516e-9, 1.675e-9, 1.883e-9,  
02262 2.101e-9, 2.349e-9, 2.614e-9, 2.92e-9, 3.305e-9, 3.724e-9,  
02263 4.142e-9, 4.887e-9, 5.614e-9, 6.506e-9, 7.463e-9, 8.817e-9,  
02264 9.849e-9, 1.187e-8, 1.321e-8, 1.474e-8, 1.698e-8, 1.794e-8,  
02265 2.09e-8, 2.211e-8, 2.362e-8, 2.556e-8, 2.729e-8, 2.88e-8,  
02266 3.046e-8, 3.167e-8, 3.367e-8, 3.457e-8, 3.59e-8, 3.711e-8,  
02267 3.826e-8, 4.001e-8, 4.211e-8, 4.315e-8, 4.661e-8, 5.01e-8,  
02268 5.249e-8, 5.84e-8, 6.628e-8, 7.512e-8, 8.253e-8, 9.722e-8,  
02269 1.067e-7, 1.153e-7, 1.347e-7, 1.428e-7, 1.577e-7, 1.694e-7,  
02270 1.833e-7, 1.938e-7, 2.108e-7, 2.059e-7, 2.157e-7, 2.185e-7,  
02271 2.208e-7, 2.182e-7, 2.093e-7, 2.014e-7, 1.962e-7, 1.819e-7,  
02272 1.713e-7, 1.51e-7, 1.34e-7, 1.154e-7, 9.89e-8, 8.88e-8, 7.673e-8,  
02273 6.599e-8, 5.73e-8, 5.081e-8, 4.567e-8, 4.147e-8, 3.773e-8,  
02274 3.46e-8, 3.194e-8, 2.953e-8, 2.759e-8, 2.594e-8, 2.442e-8,  
02275 2.355e-8, 2.283e-8, 2.279e-8, 2.231e-8, 2.279e-8, 2.239e-8,  
02276 2.21e-8, 2.309e-8, 2.293e-8, 2.352e-8, 2.415e-8, 2.43e-8,  
02277 2.426e-8, 2.465e-8, 2.5e-8, 2.496e-8, 2.465e-8, 2.445e-8,  
02278 2.383e-8, 2.299e-8, 2.165e-8, 2.113e-8, 1.968e-8, 1.819e-8,  
02279 1.644e-8, 1.427e-8, 1.27e-8, 1.082e-8, 9.428e-9, 8.091e-9,  
02280 6.958e-9, 5.988e-9, 5.246e-9, 4.601e-9, 4.098e-9, 3.664e-9,  
02281 3.287e-9, 2.942e-9, 2.656e-9, 2.364e-9, 2.118e-9, 1.903e-9,  
02282 1.703e-9, 1.525e-9, 1.365e-9, 1.229e-9, 1.107e-9, 9.96e-10,  
02283 8.945e-10, 8.08e-10, 7.308e-10, 6.616e-10, 5.994e-10, 5.422e-10,  
02284 4.929e-10, 4.478e-10, 4.07e-10, 3.707e-10, 3.379e-10, 3.087e-10,  
02285 2.823e-10, 2.592e-10, 2.385e-10, 2.201e-10, 2.038e-10, 1.897e-10,  
02286 1.774e-10, 1.667e-10, 1.577e-10, 1.502e-10, 1.437e-10, 1.394e-10,  
02287 1.358e-10, 1.324e-10, 1.329e-10, 1.324e-10, 1.36e-10, 1.39e-10,  
02288 1.424e-10, 1.544e-10, 1.651e-10, 1.817e-10, 1.984e-10, 2.195e-10,  
02289 2.438e-10, 2.7e-10, 2.991e-10, 3.322e-10, 3.632e-10, 3.957e-10,  
02290 4.36e-10, 4.701e-10, 5.03e-10, 5.381e-10, 5.793e-10, 6.19e-10,  
02291 6.596e-10, 7.004e-10, 7.561e-10, 7.934e-10, 8.552e-10, 9.142e-10,  
02292 9.57e-10, 1.027e-9, 1.097e-9, 1.193e-9, 1.334e-9, 1.47e-9,  
02293 1.636e-9, 1.871e-9, 2.122e-9, 2.519e-9, 2.806e-9, 3.203e-9,  
02294 3.846e-9, 4.362e-9, 5.114e-9, 5.643e-9, 6.305e-9, 6.981e-9,  
02295 7.983e-9, 8.783e-9, 9.419e-9, 1.017e-8, 1.063e-8, 1.121e-8,  
02296 1.13e-8, 1.201e-8, 1.225e-8, 1.232e-8, 1.223e-8, 1.177e-8,  
02297 1.151e-8, 1.116e-8, 1.047e-8, 9.698e-9, 8.734e-9, 8.202e-9,  
02298 7.041e-9, 6.074e-9, 5.172e-9, 4.468e-9, 3.913e-9, 3.414e-9,  
02299 2.975e-9, 2.65e-9, 2.406e-9, 2.173e-9, 2.009e-9, 1.861e-9,  
02300 1.727e-9, 1.612e-9, 1.514e-9, 1.43e-9, 1.362e-9, 1.333e-9,  
02301 1.288e-9, 1.249e-9, 1.238e-9, 1.228e-9, 1.217e-9, 1.202e-9,  
02302 1.209e-9, 1.177e-9, 1.157e-9, 1.165e-9, 1.142e-9, 1.131e-9,  
02303 1.138e-9, 1.117e-9, 1.1e-9, 1.069e-9, 1.023e-9, 1.005e-9,  
02304 9.159e-10, 8.863e-10, 7.865e-10, 7.153e-10, 6.247e-10, 5.846e-10,  
02305 5.133e-10, 4.36e-10, 3.789e-10, 3.335e-10, 2.833e-10, 2.483e-10,  
02306 2.155e-10, 1.918e-10, 1.709e-10, 1.529e-10, 1.374e-10, 1.235e-10,  
02307 1.108e-10, 9.933e-11, 8.932e-11, 8.022e-11, 7.224e-11, 6.52e-11,  
02308 5.896e-11, 5.328e-11, 4.813e-11, 4.365e-11, 3.961e-11, 3.594e-11,  
02309 3.266e-11, 2.967e-11, 2.701e-11, 2.464e-11, 2.248e-11, 2.054e-11,  
02310 1.878e-11, 1.721e-11, 1.579e-11, 1.453e-11, 1.341e-11, 1.241e-11,  
02311 1.154e-11, 1.078e-11, 1.014e-11, 9.601e-12, 9.167e-12, 8.838e-12,  
02312 8.614e-12, 8.493e-12, 8.481e-12, 8.581e-12, 8.795e-12, 9.131e-12,  
02313 9.601e-12, 1.021e-11, 1.097e-11, 1.191e-11, 1.303e-11, 1.439e-11,  
02314 1.601e-11, 1.778e-11, 1.984e-11, 2.234e-11, 2.474e-11, 2.766e-11,  
02315 3.085e-11, 3.415e-11, 3.821e-11, 4.261e-11, 4.748e-11, 5.323e-11,  
02316 5.935e-11, 6.619e-11, 7.418e-11, 8.294e-11, 9.26e-11, 1.039e-10,  
02317 1.156e-10, 1.297e-10, 1.46e-10, 1.641e-10, 1.858e-10, 2.1e-10,  
02318 2.383e-10, 2.724e-10, 3.116e-10, 3.538e-10, 4.173e-10, 4.727e-10,  
02319 5.503e-10, 6.337e-10, 7.32e-10, 8.298e-10, 9.328e-10, 1.059e-9,  
02320 1.176e-9, 1.328e-9, 1.445e-9, 1.593e-9, 1.77e-9, 1.954e-9,  
02321 2.175e-9, 2.405e-9, 2.622e-9, 2.906e-9, 3.294e-9, 3.713e-9,  
02322 3.98e-9, 4.384e-9, 4.987e-9, 5.311e-9, 5.874e-9, 6.337e-9,  
02323 7.027e-9, 7.39e-9, 7.769e-9, 8.374e-9, 8.605e-9, 9.165e-9,  
02324 9.415e-9, 9.511e-9, 9.704e-9, 9.588e-9, 9.45e-9, 9.086e-9,

```
02325 8.798e-9, 8.469e-9, 7.697e-9, 7.168e-9, 6.255e-9, 5.772e-9,
02326 4.97e-9, 4.271e-9, 3.653e-9, 3.154e-9, 2.742e-9, 2.435e-9,
02327 2.166e-9, 1.936e-9, 1.731e-9, 1.556e-9, 1.399e-9, 1.272e-9,
02328 1.157e-9, 1.066e-9, 9.844e-10, 9.258e-10, 8.787e-10, 8.421e-10,
02329 8.083e-10, 8.046e-10, 8.067e-10, 8.181e-10, 8.325e-10, 8.517e-10,
02330 9.151e-10, 9.351e-10, 9.677e-10, 1.071e-9, 1.126e-9, 1.219e-9,
02331 1.297e-9, 1.408e-9, 1.476e-9, 1.517e-9, 1.6e-9, 1.649e-9,
02332 1.678e-9, 1.746e-9, 1.742e-9, 1.728e-9, 1.699e-9, 1.655e-9,
02333 1.561e-9, 1.48e-9, 1.451e-9, 1.411e-9, 1.171e-9, 1.106e-9,
02334 9.714e-10, 8.523e-10, 7.346e-10, 6.241e-10, 5.371e-10, 4.704e-10,
02335 4.144e-10, 3.683e-10, 3.292e-10, 2.942e-10, 2.62e-10, 2.341e-10,
02336 2.104e-10, 1.884e-10, 1.7e-10, 1.546e-10, 1.394e-10, 1.265e-10,
02337 1.14e-10, 1.019e-10, 9.279e-11, 8.283e-11, 7.458e-11, 6.668e-11,
02338 5.976e-11, 5.33e-11, 4.794e-11, 4.289e-11, 3.841e-11, 3.467e-11,
02339 3.13e-11, 2.832e-11, 2.582e-11, 2.356e-11, 2.152e-11, 1.97e-11,
02340 1.808e-11, 1.664e-11, 1.539e-11, 1.434e-11, 1.344e-11, 1.269e-11,
02341 1.209e-11, 1.162e-11, 1.129e-11, 1.108e-11, 1.099e-11, 1.103e-11,
02342 1.119e-11, 1.148e-11, 1.193e-11, 1.252e-11, 1.329e-11, 1.421e-11,
02343 1.555e-11, 1.685e-11, 1.839e-11, 2.054e-11, 2.317e-11, 2.571e-11,
02344 2.839e-11, 3.171e-11, 3.49e-11, 3.886e-11, 4.287e-11, 4.645e-11,
02345 5.047e-11, 5.592e-11, 6.109e-11, 6.628e-11, 7.381e-11, 8.088e-11,
02346 8.966e-11, 1.045e-10, 1.12e-10, 1.287e-10, 1.486e-10, 1.662e-10,
02347 1.866e-10, 2.133e-10, 2.524e-10, 2.776e-10, 3.204e-10, 3.559e-10,
02348 4.028e-10, 4.448e-10, 4.882e-10, 5.244e-10, 5.605e-10, 6.018e-10,
02349 6.328e-10, 6.579e-10, 6.541e-10, 7.024e-10, 7.074e-10, 7.068e-10,
02350 7.009e-10, 6.698e-10, 6.545e-10, 6.209e-10, 5.834e-10, 5.412e-10,
02351 5.001e-10, 4.231e-10, 3.727e-10, 3.211e-10, 2.833e-10, 2.447e-10,
02352 2.097e-10, 1.843e-10, 1.639e-10, 1.449e-10, 1.27e-10, 1.161e-10,
02353 1.033e-10, 9.282e-11, 8.407e-11, 7.639e-11, 7.023e-11, 6.474e-11,
02354 6.142e-11, 5.76e-11, 5.568e-11, 5.472e-11, 5.39e-11, 5.455e-11,
02355 5.54e-11, 5.587e-11, 6.23e-11, 6.49e-11, 6.868e-11, 7.382e-11,
02356 8.022e-11, 8.372e-11, 9.243e-11, 1.004e-10, 1.062e-10, 1.13e-10,
02357 1.176e-10, 1.244e-10, 1.279e-10, 1.298e-10, 1.302e-10, 1.312e-10,
02358 1.295e-10, 1.244e-10, 1.211e-10, 1.167e-10, 1.098e-10, 9.927e-11,
02359 8.854e-11, 8.011e-11, 7.182e-11, 5.923e-11, 5.212e-11, 4.453e-11,
02360 3.832e-11, 3.371e-11, 2.987e-11, 2.651e-11, 2.354e-11, 2.093e-11,
02361 1.863e-11, 1.662e-11, 1.486e-11, 1.331e-11, 1.193e-11, 1.071e-11,
02362 9.628e-12, 8.66e-12, 7.801e-12, 7.031e-12, 6.347e-12, 5.733e-12,
02363 5.182e-12, 4.695e-12, 4.26e-12, 3.874e-12, 3.533e-12, 3.235e-12,
02364 2.979e-12, 2.76e-12, 2.579e-12, 2.432e-12, 2.321e-12, 2.246e-12,
02365 2.205e-12, 2.196e-12, 2.223e-12, 2.288e-12, 2.387e-12, 2.525e-12,
02366 2.704e-12, 2.925e-12, 3.191e-12, 3.508e-12, 3.876e-12, 4.303e-12,
02367 4.793e-12, 5.347e-12, 5.978e-12, 6.682e-12, 7.467e-12, 8.34e-12,
02368 9.293e-12, 1.035e-11, 1.152e-11, 1.285e-11, 1.428e-11, 1.586e-11,
02369 1.764e-11, 1.972e-11, 2.214e-11, 2.478e-11, 2.776e-11, 3.151e-11,
02370 3.591e-11, 4.103e-11, 4.66e-11, 5.395e-11, 6.306e-11, 7.172e-11,
02371 8.358e-11, 9.67e-11, 1.11e-10, 1.325e-10, 1.494e-10, 1.736e-10,
02372 2.007e-10, 2.296e-10, 2.608e-10, 3.004e-10, 3.361e-10, 3.727e-10,
02373 4.373e-10, 4.838e-10, 5.483e-10, 6.006e-10, 6.535e-10, 6.899e-10,
02374 7.687e-10, 8.444e-10, 8.798e-10, 9.135e-10, 9.532e-10, 9.757e-10,
02375 9.968e-10, 1.006e-9, 9.949e-10, 9.789e-10, 9.564e-10, 9.215e-10,
02376 8.51e-10, 8.394e-10, 7.707e-10, 7.152e-10, 6.274e-10, 5.598e-10,
02377 5.028e-10, 4.3e-10, 3.71e-10, 3.245e-10, 2.809e-10, 2.461e-10,
02378 2.154e-10, 1.91e-10, 1.685e-10, 1.487e-10, 1.313e-10, 1.163e-10,
02379 1.031e-10, 9.172e-11, 8.221e-11, 7.382e-11, 6.693e-11, 6.079e-11,
02380 5.581e-11, 5.167e-11, 4.811e-11, 4.506e-11, 4.255e-11, 4.083e-11,
02381 3.949e-11, 3.881e-11, 3.861e-11, 3.858e-11, 3.951e-11, 4.045e-11,
02382 4.24e-11, 4.487e-11, 4.806e-11, 5.133e-11, 5.518e-11, 5.919e-11,
02383 6.533e-11, 7.031e-11, 7.762e-11, 8.305e-11, 9.252e-11, 9.727e-11,
02384 1.045e-10, 1.117e-10, 1.2e-10, 1.275e-10, 1.341e-10, 1.362e-10,
02385 1.438e-10, 1.45e-10, 1.455e-10, 1.455e-10, 1.434e-10, 1.381e-10,
02386 1.301e-10, 1.276e-10, 1.163e-10, 1.089e-10, 9.911e-11, 8.943e-11,
02387 7.618e-11, 6.424e-11, 5.717e-11, 4.866e-11, 4.257e-11, 3.773e-11,
02388 3.331e-11, 2.958e-11, 2.629e-11, 2.316e-11, 2.073e-11, 1.841e-11,
02389 1.635e-11, 1.464e-11, 1.31e-11, 1.16e-11, 1.047e-11, 9.408e-12,
02390 8.414e-12, 7.521e-12, 6.705e-12, 5.993e-12, 5.371e-12, 4.815e-12,
02391 4.338e-12, 3.921e-12, 3.567e-12, 3.265e-12, 3.01e-12, 2.795e-12,
02392 2.613e-12, 2.464e-12, 2.346e-12, 2.256e-12, 2.195e-12, 2.165e-12,
02393 2.166e-12, 2.198e-12, 2.262e-12, 2.364e-12, 2.502e-12, 2.682e-12,
02394 2.908e-12, 3.187e-12, 3.533e-12, 3.946e-12, 4.418e-12, 5.013e-12,
02395 5.708e-12, 6.379e-12, 7.43e-12, 8.39e-12, 9.51e-12, 1.078e-11,
02396 1.259e-11, 1.438e-11, 1.63e-11, 1.814e-11, 2.055e-11, 2.348e-11,
02397 2.664e-11, 2.956e-11, 3.3e-11, 3.677e-11, 4.032e-11, 4.494e-11,
02398 4.951e-11, 5.452e-11, 6.014e-11, 6.5e-11, 6.915e-11, 7.45e-11,
02399 7.971e-11, 8.468e-11, 8.726e-11, 8.995e-11, 9.182e-11, 9.509e-11,
02400 9.333e-11, 9.386e-11, 9.457e-11, 9.21e-11, 9.019e-11, 8.68e-11,
02401 8.298e-11, 7.947e-11, 7.46e-11, 7.082e-11, 6.132e-11, 5.855e-11,
02402 5.073e-11, 4.464e-11, 3.825e-11, 3.375e-11, 2.911e-11, 2.535e-11,
02403 2.16e-11, 1.907e-11, 1.665e-11, 1.463e-11, 1.291e-11, 1.133e-11,
02404 9.997e-12, 8.836e-12, 7.839e-12, 6.943e-12, 6.254e-12, 5.6e-12,
02405 5.029e-12, 4.529e-12, 4.102e-12, 3.737e-12, 3.428e-12, 3.169e-12,
02406 2.959e-12, 2.798e-12, 2.675e-12, 2.582e-12, 2.644e-12, 2.557e-12,
02407 2.614e-12, 2.717e-12, 2.874e-12, 3.056e-12, 3.187e-12, 3.631e-12,
02408 3.979e-12, 4.248e-12, 4.817e-12, 5.266e-12, 5.836e-12, 6.365e-12,
02409 6.807e-12, 7.47e-12, 7.951e-12, 8.636e-12, 8.972e-12, 9.314e-12,
02410 9.445e-12, 1.003e-11, 1.013e-11, 9.937e-12, 9.729e-12, 9.064e-12,
02411 9.119e-12, 9.124e-12, 8.704e-12, 8.078e-12, 7.47e-12, 6.329e-12,
```



```
02412    5.674e-12, 4.808e-12, 4.119e-12, 3.554e-12, 3.103e-12, 2.731e-12,
02413    2.415e-12, 2.15e-12, 1.926e-12, 1.737e-12, 1.578e-12, 1.447e-12,
02414    1.34e-12, 1.255e-12, 1.191e-12, 1.146e-12, 1.121e-12, 1.114e-12,
02415    1.126e-12, 1.156e-12, 1.207e-12, 1.278e-12, 1.372e-12, 1.49e-12,
02416    1.633e-12, 1.805e-12, 2.01e-12, 2.249e-12, 2.528e-12, 2.852e-12,
02417    3.228e-12, 3.658e-12, 4.153e-12, 4.728e-12, 5.394e-12, 6.176e-12,
02418    7.126e-12, 8.188e-12, 9.328e-12, 1.103e-11, 1.276e-11, 1.417e-11,
02419    1.615e-11, 1.84e-11, 2.155e-11, 2.429e-11, 2.826e-11, 3.222e-11,
02420    3.664e-11, 4.14e-11, 4.906e-11, 5.536e-11, 6.327e-11, 7.088e-11,
02421    8.316e-11, 9.242e-11, 1.07e-10, 1.223e-10, 1.341e-10, 1.553e-10,
02422    1.703e-10, 1.9e-10, 2.022e-10, 2.233e-10, 2.345e-10, 2.438e-10,
02423    2.546e-10, 2.599e-10, 2.661e-10, 2.703e-10, 2.686e-10, 2.662e-10,
02424    2.56e-10, 2.552e-10, 2.378e-10, 2.252e-10, 2.146e-10, 1.885e-10,
02425    1.668e-10, 1.441e-10, 1.295e-10, 1.119e-10, 9.893e-11, 8.687e-11,
02426    7.678e-11, 6.685e-11, 5.879e-11, 5.127e-11, 4.505e-11, 3.997e-11,
02427    3.511e-11
02428    };
02429
02430    static double h2ofrn[2001] = { .01095, .01126, .01205, .01322, .0143,
02431    .01506, .01548, .01534, .01486, .01373, .01262, .01134, .01001,
02432    .008702, .007475, .006481, .00548, .0046, .003833, .00311,
02433    .002543, .002049, .00168, .001374, .001046, 8.193e-4, 6.267e-4,
02434    4.968e-4, 3.924e-4, 2.983e-4, 2.477e-4, 1.997e-4, 1.596e-4,
02435    1.331e-4, 1.061e-4, 8.942e-5, 7.168e-5, 5.887e-5, 4.848e-5,
02436    3.817e-5, 3.17e-5, 2.579e-5, 2.162e-5, 1.768e-5, 1.49e-5,
02437    1.231e-5, 1.013e-5, 8.555e-6, 7.328e-6, 6.148e-6, 5.207e-6,
02438    4.387e-6, 3.741e-6, 3.22e-6, 2.753e-6, 2.346e-6, 1.985e-6,
02439    1.716e-6, 1.475e-6, 1.286e-6, 1.122e-6, 9.661e-7, 8.284e-7,
02440    7.057e-7, 6.119e-7, 5.29e-7, 4.571e-7, 3.948e-7, 3.432e-7,
02441    2.983e-7, 2.589e-7, 2.265e-7, 1.976e-7, 1.704e-7, 1.456e-7,
02442    1.26e-7, 1.101e-7, 9.648e-8, 8.415e-8, 7.34e-8, 6.441e-8,
02443    5.643e-8, 4.94e-8, 4.276e-8, 3.703e-8, 3.227e-8, 2.825e-8,
02444    2.478e-8, 2.174e-8, 1.898e-8, 1.664e-8, 1.458e-8, 1.278e-8,
02445    1.126e-8, 9.891e-9, 8.709e-9, 7.652e-9, 6.759e-9, 5.975e-9,
02446    5.31e-9, 4.728e-9, 4.214e-9, 3.792e-9, 3.463e-9, 3.226e-9,
02447    2.992e-9, 2.813e-9, 2.749e-9, 2.809e-9, 2.913e-9, 3.037e-9,
02448    3.413e-9, 3.738e-9, 4.189e-9, 4.808e-9, 5.978e-9, 7.088e-9,
02449    8.071e-9, 9.61e-9, 1.21e-8, 1.5e-8, 1.764e-8, 2.221e-8, 2.898e-8,
02450    3.948e-8, 5.068e-8, 6.227e-8, 7.898e-8, 1.033e-7, 1.437e-7,
02451    1.889e-7, 2.589e-7, 3.59e-7, 4.971e-7, 7.156e-7, 9.983e-7,
02452    1.381e-6, 1.929e-6, 2.591e-6, 3.453e-6, 4.57e-6, 5.93e-6,
02453    7.552e-6, 9.556e-6, 1.183e-5, 1.425e-5, 1.681e-5, 1.978e-5,
02454    2.335e-5, 2.668e-5, 3.022e-5, 3.371e-5, 3.715e-5, 3.967e-5,
02455    4.06e-5, 4.01e-5, 3.809e-5, 3.491e-5, 3.155e-5, 2.848e-5,
02456    2.678e-5, 2.66e-5, 2.811e-5, 3.071e-5, 3.294e-5, 3.459e-5,
02457    3.569e-5, 3.56e-5, 3.434e-5, 3.186e-5, 2.916e-5, 2.622e-5,
02458    2.275e-5, 1.918e-5, 1.62e-5, 1.373e-5, 1.182e-5, 1.006e-5,
02459    8.556e-6, 7.26e-6, 6.107e-6, 5.034e-6, 4.211e-6, 3.426e-6,
02460    2.865e-6, 2.446e-6, 1.998e-6, 1.628e-6, 1.242e-6, 1.005e-6,
02461    7.853e-7, 6.21e-7, 5.071e-7, 4.156e-7, 3.548e-7, 2.825e-7,
02462    2.261e-7, 1.916e-7, 1.51e-7, 1.279e-7, 1.059e-7, 9.14e-8,
02463    7.707e-8, 6.17e-8, 5.311e-8, 4.263e-8, 3.518e-8, 2.961e-8,
02464    2.457e-8, 2.119e-8, 1.712e-8, 1.439e-8, 1.201e-8, 1.003e-8,
02465    8.564e-9, 7.199e-9, 6.184e-9, 5.206e-9, 4.376e-9, 3.708e-9,
02466    3.157e-9, 2.725e-9, 2.361e-9, 2.074e-9, 1.797e-9, 1.562e-9,
02467    1.364e-9, 1.196e-9, 1.042e-9, 8.862e-10, 7.648e-10, 6.544e-10,
02468    5.609e-10, 4.791e-10, 4.108e-10, 3.531e-10, 3.038e-10, 2.618e-10,
02469    2.268e-10, 1.969e-10, 1.715e-10, 1.496e-10, 1.308e-10, 1.147e-10,
02470    1.008e-10, 8.894e-11, 7.885e-11, 7.031e-11, 6.355e-11, 5.854e-11,
02471    5.534e-11, 5.466e-11, 5.725e-11, 6.447e-11, 7.943e-11, 1.038e-10,
02472    1.437e-10, 2.04e-10, 2.901e-10, 4.051e-10, 5.556e-10, 7.314e-10,
02473    9.291e-10, 1.134e-9, 1.321e-9, 1.482e-9, 1.596e-9, 1.669e-9,
02474    1.715e-9, 1.762e-9, 1.817e-9, 1.828e-9, 1.848e-9, 1.873e-9,
02475    1.902e-9, 1.894e-9, 1.864e-9, 1.841e-9, 1.797e-9, 1.704e-9,
02476    1.559e-9, 1.382e-9, 1.187e-9, 1.001e-9, 8.468e-10, 7.265e-10,
02477    6.521e-10, 6.381e-10, 6.66e-10, 7.637e-10, 9.705e-10, 1.368e-9,
02478    1.856e-9, 2.656e-9, 3.954e-9, 5.96e-9, 8.72e-9, 1.247e-8,
02479    1.781e-8, 2.491e-8, 3.311e-8, 4.272e-8, 5.205e-8, 6.268e-8,
02480    7.337e-8, 8.277e-8, 9.185e-8, 1.004e-7, 1.091e-7, 1.159e-7,
02481    1.188e-7, 1.175e-7, 1.124e-7, 1.033e-7, 9.381e-8, 8.501e-8,
02482    7.956e-8, 7.894e-8, 8.331e-8, 9.102e-8, 9.836e-8, 1.035e-7,
02483    1.064e-7, 1.06e-7, 1.032e-7, 9.808e-8, 9.139e-8, 8.442e-8,
02484    7.641e-8, 6.881e-8, 6.161e-8, 5.404e-8, 4.804e-8, 4.446e-8,
02485    4.328e-8, 4.259e-8, 4.421e-8, 4.673e-8, 4.985e-8, 5.335e-8,
02486    5.796e-8, 6.542e-8, 7.714e-8, 8.827e-8, 1.04e-7, 1.238e-7,
02487    1.499e-7, 1.829e-7, 2.222e-7, 2.689e-7, 3.303e-7, 3.981e-7,
02488    4.84e-7, 5.91e-7, 7.363e-7, 9.087e-7, 1.139e-6, 1.455e-6,
02489    1.866e-6, 2.44e-6, 3.115e-6, 3.941e-6, 4.891e-6, 5.992e-6,
02490    7.111e-6, 8.296e-6, 9.21e-6, 9.987e-6, 1.044e-5, 1.073e-5,
02491    1.092e-5, 1.106e-5, 1.138e-5, 1.171e-5, 1.186e-5, 1.186e-5,
02492    1.179e-5, 1.166e-5, 1.151e-5, 1.16e-5, 1.197e-5, 1.241e-5,
02493    1.268e-5, 1.26e-5, 1.184e-5, 1.063e-5, 9.204e-6, 7.584e-6,
02494    6.053e-6, 4.482e-6, 3.252e-6, 2.337e-6, 1.662e-6, 1.18e-6,
02495    8.15e-7, 5.95e-7, 4.354e-7, 3.302e-7, 2.494e-7, 1.93e-7,
02496    1.545e-7, 1.25e-7, 1.039e-7, 8.602e-8, 7.127e-8, 5.897e-8,
02497    4.838e-8, 4.018e-8, 3.28e-8, 2.72e-8, 2.307e-8, 1.972e-8,
02498    1.654e-8, 1.421e-8, 1.174e-8, 1.004e-8, 8.739e-9, 7.358e-9,
```

```
02499 6.242e-9, 5.303e-9, 4.567e-9, 3.94e-9, 3.375e-9, 2.864e-9,
02500 2.422e-9, 2.057e-9, 1.75e-9, 1.505e-9, 1.294e-9, 1.101e-9,
02501 9.401e-10, 8.018e-10, 6.903e-10, 5.965e-10, 5.087e-10, 4.364e-10,
02502 3.759e-10, 3.247e-10, 2.809e-10, 2.438e-10, 2.123e-10, 1.853e-10,
02503 1.622e-10, 1.426e-10, 1.26e-10, 1.125e-10, 1.022e-10, 9.582e-11,
02504 9.388e-11, 9.801e-11, 1.08e-10, 1.276e-10, 1.551e-10, 1.903e-10,
02505 2.291e-10, 2.724e-10, 3.117e-10, 3.4e-10, 3.562e-10, 3.625e-10,
02506 3.619e-10, 3.429e-10, 3.221e-10, 2.943e-10, 2.645e-10, 2.338e-10,
02507 2.062e-10, 1.901e-10, 1.814e-10, 1.827e-10, 1.906e-10, 1.984e-10,
02508 2.04e-10, 2.068e-10, 2.075e-10, 2.018e-10, 1.959e-10, 1.897e-10,
02509 1.852e-10, 1.791e-10, 1.696e-10, 1.634e-10, 1.598e-10, 1.561e-10,
02510 1.518e-10, 1.443e-10, 1.377e-10, 1.346e-10, 1.342e-10, 1.375e-10,
02511 1.525e-10, 1.767e-10, 2.108e-10, 2.524e-10, 2.981e-10, 3.477e-10,
02512 4.262e-10, 5.326e-10, 6.646e-10, 8.321e-10, 1.069e-9, 1.386e-9,
02513 1.743e-9, 2.216e-9, 2.808e-9, 3.585e-9, 4.552e-9, 5.907e-9,
02514 7.611e-9, 9.774e-9, 1.255e-8, 1.666e-8, 2.279e-8, 3.221e-8,
02515 4.531e-8, 6.4e-8, 9.187e-8, 1.295e-7, 1.825e-7, 2.431e-7,
02516 3.181e-7, 4.009e-7, 4.941e-7, 5.88e-7, 6.623e-7, 7.155e-7,
02517 7.451e-7, 7.594e-7, 7.541e-7, 7.467e-7, 7.527e-7, 7.935e-7,
02518 8.461e-7, 8.954e-7, 9.364e-7, 9.843e-7, 1.024e-6, 1.05e-6,
02519 1.059e-6, 1.074e-6, 1.072e-6, 1.043e-6, 9.789e-7, 8.803e-7,
02520 7.662e-7, 6.378e-7, 5.133e-7, 3.958e-7, 2.914e-7, 2.144e-7,
02521 1.57e-7, 1.14e-7, 8.47e-8, 6.2e-8, 4.657e-8, 3.559e-8, 2.813e-8,
02522 2.222e-8, 1.769e-8, 1.391e-8, 1.125e-8, 9.186e-9, 7.704e-9,
02523 6.447e-9, 5.381e-9, 4.442e-9, 3.669e-9, 3.057e-9, 2.564e-9,
02524 2.153e-9, 1.784e-9, 1.499e-9, 1.281e-9, 1.082e-9, 9.304e-10,
02525 8.169e-10, 6.856e-10, 5.866e-10, 5.043e-10, 4.336e-10, 3.731e-10,
02526 3.175e-10, 2.745e-10, 2.374e-10, 2.007e-10, 1.737e-10, 1.508e-10,
02527 1.302e-10, 1.13e-10, 9.672e-11, 8.375e-11, 7.265e-11, 6.244e-11,
02528 5.343e-11, 4.654e-11, 3.975e-11, 3.488e-11, 3.097e-11, 2.834e-11,
02529 2.649e-11, 2.519e-11, 2.462e-11, 2.443e-11, 2.44e-11, 2.398e-11,
02530 2.306e-11, 2.183e-11, 2.021e-11, 1.821e-11, 1.599e-11, 1.403e-11,
02531 1.196e-11, 1.023e-11, 8.728e-12, 7.606e-12, 6.941e-12, 6.545e-12,
02532 6.484e-12, 6.6e-12, 6.718e-12, 6.785e-12, 6.746e-12, 6.724e-12,
02533 6.764e-12, 6.995e-12, 7.144e-12, 7.32e-12, 7.33e-12, 7.208e-12,
02534 6.789e-12, 6.09e-12, 5.337e-12, 4.62e-12, 4.037e-12, 3.574e-12,
02535 3.311e-12, 3.346e-12, 3.566e-12, 3.836e-12, 4.076e-12, 4.351e-12,
02536 4.691e-12, 5.114e-12, 5.427e-12, 6.167e-12, 7.436e-12, 8.842e-12,
02537 1.038e-11, 1.249e-11, 1.54e-11, 1.915e-11, 2.48e-11, 3.256e-11,
02538 4.339e-11, 5.611e-11, 7.519e-11, 1.037e-10, 1.409e-10, 1.883e-10,
02539 2.503e-10, 3.38e-10, 4.468e-10, 5.801e-10, 7.335e-10, 8.98e-10,
02540 1.11e-9, 1.363e-9, 1.677e-9, 2.104e-9, 2.681e-9, 3.531e-9,
02541 4.621e-9, 6.106e-9, 8.154e-9, 1.046e-8, 1.312e-8, 1.607e-8,
02542 1.948e-8, 2.266e-8, 2.495e-8, 2.655e-8, 2.739e-8, 2.739e-8,
02543 2.662e-8, 2.589e-8, 2.59e-8, 2.664e-8, 2.833e-8, 3.023e-8,
02544 3.305e-8, 3.558e-8, 3.793e-8, 3.961e-8, 4.056e-8, 4.102e-8,
02545 4.025e-8, 3.917e-8, 3.706e-8, 3.493e-8, 3.249e-8, 3.096e-8,
02546 3.011e-8, 3.111e-8, 3.395e-8, 3.958e-8, 4.875e-8, 6.066e-8,
02547 7.915e-8, 1.011e-7, 1.3e-7, 1.622e-7, 2.003e-7, 2.448e-7,
02548 2.863e-7, 3.317e-7, 3.655e-7, 3.96e-7, 4.098e-7, 4.168e-7,
02549 4.198e-7, 4.207e-7, 4.289e-7, 4.384e-7, 4.471e-7, 4.524e-7,
02550 4.574e-7, 4.633e-7, 4.785e-7, 5.028e-7, 5.371e-7, 5.727e-7,
02551 5.955e-7, 5.998e-7, 5.669e-7, 5.082e-7, 4.397e-7, 3.596e-7,
02552 2.814e-7, 2.074e-7, 1.486e-7, 1.057e-7, 7.25e-8, 4.946e-8,
02553 3.43e-8, 2.447e-8, 1.793e-8, 1.375e-8, 1.096e-8, 9.091e-9,
02554 7.709e-9, 6.631e-9, 5.714e-9, 4.886e-9, 4.205e-9, 3.575e-9,
02555 3.07e-9, 2.631e-9, 2.284e-9, 2.002e-9, 1.745e-9, 1.509e-9,
02556 1.284e-9, 1.084e-9, 9.163e-10, 7.663e-10, 6.346e-10, 5.283e-10,
02557 4.354e-10, 3.59e-10, 2.982e-10, 2.455e-10, 2.033e-10, 1.696e-10,
02558 1.432e-10, 1.211e-10, 1.02e-10, 8.702e-11, 7.38e-11, 6.293e-11,
02559 5.343e-11, 4.532e-11, 3.907e-11, 3.365e-11, 2.945e-11, 2.558e-11,
02560 2.192e-11, 1.895e-11, 1.636e-11, 1.42e-11, 1.228e-11, 1.063e-11,
02561 9.348e-12, 8.2e-12, 7.231e-12, 6.43e-12, 5.702e-12, 5.052e-12,
02562 4.469e-12, 4e-12, 3.679e-12, 3.387e-12, 3.197e-12, 3.158e-12,
02563 3.327e-12, 3.675e-12, 4.292e-12, 5.437e-12, 7.197e-12, 1.008e-11,
02564 1.437e-11, 2.035e-11, 2.905e-11, 4.062e-11, 5.528e-11, 7.177e-11,
02565 9.064e-11, 1.109e-10, 1.297e-10, 1.473e-10, 1.652e-10, 1.851e-10,
02566 2.079e-10, 2.313e-10, 2.619e-10, 2.958e-10, 3.352e-10, 3.796e-10,
02567 4.295e-10, 4.923e-10, 5.49e-10, 5.998e-10, 6.388e-10, 6.645e-10,
02568 6.712e-10, 6.549e-10, 6.38e-10, 6.255e-10, 6.253e-10, 6.459e-10,
02569 6.977e-10, 7.59e-10, 8.242e-10, 8.92e-10, 9.403e-10, 9.701e-10,
02570 9.483e-10, 9.135e-10, 8.617e-10, 7.921e-10, 7.168e-10, 6.382e-10,
02571 5.677e-10, 5.045e-10, 4.572e-10, 4.312e-10, 4.145e-10, 4.192e-10,
02572 4.541e-10, 5.368e-10, 6.771e-10, 8.962e-10, 1.21e-9, 1.659e-9,
02573 2.33e-9, 3.249e-9, 4.495e-9, 5.923e-9, 7.642e-9, 9.607e-9,
02574 1.178e-8, 1.399e-8, 1.584e-8, 1.73e-8, 1.816e-8, 1.87e-8,
02575 1.868e-8, 1.87e-8, 1.884e-8, 1.99e-8, 2.15e-8, 2.258e-8,
02576 2.364e-8, 2.473e-8, 2.602e-8, 2.689e-8, 2.731e-8, 2.816e-8,
02577 2.859e-8, 2.839e-8, 2.703e-8, 2.451e-8, 2.149e-8, 1.787e-8,
02578 1.449e-8, 1.111e-8, 8.282e-9, 6.121e-9, 4.494e-9, 3.367e-9,
02579 2.487e-9, 1.885e-9, 1.503e-9, 1.249e-9, 1.074e-9, 9.427e-10,
02580 8.439e-10, 7.563e-10, 6.772e-10, 6.002e-10, 5.254e-10, 4.588e-10,
02581 3.977e-10, 3.449e-10, 3.003e-10, 2.624e-10, 2.335e-10, 2.04e-10,
02582 1.771e-10, 1.534e-10, 1.296e-10, 1.097e-10, 9.173e-11, 7.73e-11,
02583 6.547e-11, 5.191e-11, 4.198e-11, 3.361e-11, 2.732e-11, 2.244e-11,
02584 1.791e-11, 1.509e-11, 1.243e-11, 1.035e-11, 8.969e-12, 7.394e-12,
02585 6.323e-12, 5.282e-12, 4.543e-12, 3.752e-12, 3.14e-12, 2.6e-12,
```

02586 2.194e-12, 1.825e-12, 1.511e-12, 1.245e-12, 1.024e-12, 8.539e-13,  
02587 7.227e-13, 6.102e-13, 5.189e-13, 4.43e-13, 3.774e-13, 3.236e-13,  
02588 2.8e-13, 2.444e-13, 2.156e-13, 1.932e-13, 1.775e-13, 1.695e-13,  
02589 1.672e-13, 1.704e-13, 1.825e-13, 2.087e-13, 2.614e-13, 3.377e-13,  
02590 4.817e-13, 6.989e-13, 1.062e-12, 1.562e-12, 2.288e-12, 3.295e-12,  
02591 4.55e-12, 5.965e-12, 7.546e-12, 9.395e-12, 1.103e-11, 1.228e-11,  
02592 1.318e-11, 1.38e-11, 1.421e-11, 1.39e-11, 1.358e-11, 1.336e-11,  
02593 1.342e-11, 1.356e-11, 1.424e-11, 1.552e-11, 1.73e-11, 1.951e-11,  
02594 2.128e-11, 2.249e-11, 2.277e-11, 2.226e-11, 2.111e-11, 1.922e-11,  
02595 1.775e-11, 1.661e-11, 1.547e-11, 1.446e-11, 1.323e-11, 1.21e-11,  
02596 1.054e-11, 9.283e-12, 8.671e-12, 8.67e-12, 9.429e-12, 1.062e-11,  
02597 1.255e-11, 1.506e-11, 1.818e-11, 2.26e-11, 2.831e-11, 3.723e-11,  
02598 5.092e-11, 6.968e-11, 9.826e-11, 1.349e-10, 1.87e-10, 2.58e-10,  
02599 3.43e-10, 4.424e-10, 5.521e-10, 6.812e-10, 8.064e-10, 9.109e-10,  
02600 9.839e-10, 1.028e-9, 1.044e-9, 1.029e-9, 1.005e-9, 1.002e-9,  
02601 1.038e-9, 1.122e-9, 1.233e-9, 1.372e-9, 1.524e-9, 1.665e-9,  
02602 1.804e-9, 1.908e-9, 2.015e-9, 2.117e-9, 2.219e-9, 2.336e-9,  
02603 2.531e-9, 2.805e-9, 3.189e-9, 3.617e-9, 4.208e-9, 4.911e-9,  
02604 5.619e-9, 6.469e-9, 7.188e-9, 7.957e-9, 8.503e-9, 9.028e-9,  
02605 9.571e-9, 9.99e-9, 1.055e-8, 1.102e-8, 1.132e-8, 1.141e-8,  
02606 1.145e-8, 1.145e-8, 1.176e-8, 1.224e-8, 1.304e-8, 1.388e-8,  
02607 1.445e-8, 1.453e-8, 1.368e-8, 1.22e-8, 1.042e-8, 8.404e-9,  
02608 6.403e-9, 4.643e-9, 3.325e-9, 2.335e-9, 1.638e-9, 1.19e-9,  
02609 9.161e-10, 7.412e-10, 6.226e-10, 5.516e-10, 5.068e-10, 4.831e-10,  
02610 4.856e-10, 5.162e-10, 5.785e-10, 6.539e-10, 7.485e-10, 8.565e-10,  
02611 9.534e-10, 1.052e-9, 1.115e-9, 1.173e-9, 1.203e-9, 1.224e-9,  
02612 1.243e-9, 1.248e-9, 1.261e-9, 1.265e-9, 1.25e-9, 1.217e-9,  
02613 1.176e-9, 1.145e-9, 1.153e-9, 1.199e-9, 1.278e-9, 1.366e-9,  
02614 1.426e-9, 1.444e-9, 1.365e-9, 1.224e-9, 1.051e-9, 8.539e-10,  
02615 6.564e-10, 4.751e-10, 3.404e-10, 2.377e-10, 1.631e-10, 1.114e-10,  
02616 7.87e-11, 5.793e-11, 4.284e-11, 3.3e-11, 2.62e-11, 2.152e-11,  
02617 1.777e-11, 1.496e-11, 1.242e-11, 1.037e-11, 8.725e-12, 7.004e-12,  
02618 5.718e-12, 4.769e-12, 3.952e-12, 3.336e-12, 2.712e-12, 2.213e-12,  
02619 1.803e-12, 1.492e-12, 1.236e-12, 1.006e-12, 8.384e-13, 7.063e-13,  
02620 5.879e-13, 4.93e-13, 4.171e-13, 3.569e-13, 3.083e-13, 2.688e-13,  
02621 2.333e-13, 2.035e-13, 1.82e-13, 1.682e-13, 1.635e-13, 1.674e-13,  
02622 1.769e-13, 2.022e-13, 2.485e-13, 3.127e-13, 4.25e-13, 5.928e-13,  
02623 8.514e-13, 1.236e-12, 1.701e-12, 2.392e-12, 3.231e-12, 4.35e-12,  
02624 5.559e-12, 6.915e-12, 8.519e-12, 1.013e-11, 1.146e-11, 1.24e-11,  
02625 1.305e-11, 1.333e-11, 1.318e-11, 1.263e-11, 1.238e-11, 1.244e-11,  
02626 1.305e-11, 1.432e-11, 1.623e-11, 1.846e-11, 2.09e-11, 2.328e-11,  
02627 2.526e-11, 2.637e-11, 2.702e-11, 2.794e-11, 2.889e-11, 2.989e-11,  
02628 3.231e-11, 3.68e-11, 4.375e-11, 5.504e-11, 7.159e-11, 9.502e-11,  
02629 1.279e-10, 1.645e-10, 2.098e-10, 2.618e-10, 3.189e-10, 3.79e-10,  
02630 4.303e-10, 4.753e-10, 5.027e-10, 5.221e-10, 5.293e-10, 5.346e-10,  
02631 5.467e-10, 5.796e-10, 6.2e-10, 6.454e-10, 6.705e-10, 6.925e-10,  
02632 7.233e-10, 7.35e-10, 7.538e-10, 7.861e-10, 8.077e-10, 8.132e-10,  
02633 7.749e-10, 7.036e-10, 6.143e-10, 5.093e-10, 4.089e-10, 3.092e-10,  
02634 2.299e-10, 1.705e-10, 1.277e-10, 9.723e-11, 7.533e-11, 6.126e-11,  
02635 5.154e-11, 4.428e-11, 3.913e-11, 3.521e-11, 3.297e-11, 3.275e-11,  
02636 3.46e-11, 3.798e-11, 4.251e-11, 4.745e-11, 5.232e-11, 5.606e-11,  
02637 5.82e-11, 5.88e-11, 5.79e-11, 5.661e-11, 5.491e-11, 5.366e-11,  
02638 5.341e-11, 5.353e-11, 5.336e-11, 5.293e-11, 5.248e-11, 5.235e-11,  
02639 5.208e-11, 5.322e-11, 5.521e-11, 5.725e-11, 5.827e-11, 5.685e-11,  
02640 5.245e-11, 4.612e-11, 3.884e-11, 3.129e-11, 2.404e-11, 1.732e-11,  
02641 1.223e-11, 8.574e-12, 5.888e-12, 3.986e-12, 2.732e-12, 1.948e-12,  
02642 1.414e-12, 1.061e-12, 8.298e-13, 6.612e-13, 5.413e-13, 4.472e-13,  
02643 3.772e-13, 3.181e-13, 2.645e-13, 2.171e-13, 1.778e-13, 1.464e-13,  
02644 1.183e-13, 9.637e-14, 7.991e-14, 6.668e-14, 5.57e-14, 4.663e-14,  
02645 3.848e-14, 3.233e-14, 2.706e-14, 2.284e-14, 1.944e-14, 1.664e-14,  
02646 1.43e-14, 1.233e-14, 1.066e-14, 9.234e-15, 8.023e-15, 6.993e-15,  
02647 6.119e-15, 5.384e-15, 4.774e-15, 4.283e-15, 3.916e-15, 3.695e-15,  
02648 3.682e-15, 4.004e-15, 4.912e-15, 6.853e-15, 1.056e-14, 1.712e-14,  
02649 2.804e-14, 4.516e-14, 7.113e-14, 1.084e-13, 1.426e-13, 1.734e-13,  
02650 1.978e-13, 2.194e-13, 2.388e-13, 2.489e-13, 2.626e-13, 2.865e-13,  
02651 3.105e-13, 3.387e-13, 3.652e-13, 3.984e-13, 4.398e-13, 4.906e-13,  
02652 5.55e-13, 6.517e-13, 7.813e-13, 9.272e-13, 1.164e-12, 1.434e-12,  
02653 1.849e-12, 2.524e-12, 3.328e-12, 4.523e-12, 6.108e-12, 8.207e-12,  
02654 1.122e-11, 1.477e-11, 1.9e-11, 2.412e-11, 2.984e-11, 3.68e-11,  
02655 4.353e-11, 4.963e-11, 5.478e-11, 5.903e-11, 6.233e-11, 6.483e-11,  
02656 6.904e-11, 7.569e-11, 8.719e-11, 1.048e-10, 1.278e-10, 1.557e-10,  
02657 1.869e-10, 2.218e-10, 2.61e-10, 2.975e-10, 3.371e-10, 3.746e-10,  
02658 4.065e-10, 4.336e-10, 4.503e-10, 4.701e-10, 4.8e-10, 4.917e-10,  
02659 5.038e-10, 5.128e-10, 5.143e-10, 5.071e-10, 5.019e-10, 5.025e-10,  
02660 5.183e-10, 5.496e-10, 5.877e-10, 6.235e-10, 6.42e-10, 6.234e-10,  
02661 5.698e-10, 4.916e-10, 4.022e-10, 3.126e-10, 2.282e-10, 1.639e-10,  
02662 1.142e-10, 7.919e-11, 5.69e-11, 4.313e-11, 3.413e-11, 2.807e-11,  
02663 2.41e-11, 2.166e-11, 2.024e-11, 1.946e-11, 1.929e-11, 1.963e-11,  
02664 2.035e-11, 2.162e-11, 2.305e-11, 2.493e-11, 2.748e-11, 3.048e-11,  
02665 3.413e-11, 3.754e-11, 4.155e-11, 4.635e-11, 5.11e-11, 5.734e-11,  
02666 6.338e-11, 6.99e-11, 7.611e-11, 8.125e-11, 8.654e-11, 8.951e-11,  
02667 9.182e-11, 9.31e-11, 9.273e-11, 9.094e-11, 8.849e-11, 8.662e-11,  
02668 8.67e-11, 8.972e-11, 9.566e-11, 1.025e-10, 1.083e-10, 1.111e-10,  
02669 1.074e-10, 9.771e-11, 8.468e-11, 6.958e-11, 5.47e-11, 4.04e-11,  
02670 2.94e-11, 2.075e-11, 1.442e-11, 1.01e-11, 7.281e-12, 5.409e-12,  
02671 4.138e-12, 3.304e-12, 2.784e-12, 2.473e-12, 2.273e-12, 2.186e-12,  
02672 2.118e-12, 2.066e-12, 1.958e-12, 1.818e-12, 1.675e-12, 1.509e-12,

```
02673 1.349e-12, 1.171e-12, 9.838e-13, 8.213e-13, 6.765e-13, 5.378e-13,
02674 4.161e-13, 3.119e-13, 2.279e-13, 1.637e-13, 1.152e-13, 8.112e-14,
02675 5.919e-14, 4.47e-14, 3.492e-14, 2.811e-14, 2.319e-14, 1.948e-14,
02676 1.66e-14, 1.432e-14, 1.251e-14, 1.109e-14, 1.006e-14, 9.45e-15,
02677 9.384e-15, 1.012e-14, 1.216e-14, 1.636e-14, 2.305e-14, 3.488e-14,
02678 5.572e-14, 8.479e-14, 1.265e-13, 1.905e-13, 2.73e-13, 3.809e-13,
02679 4.955e-13, 6.303e-13, 7.861e-13, 9.427e-13, 1.097e-12, 1.212e-12,
02680 1.328e-12, 1.415e-12, 1.463e-12, 1.495e-12, 1.571e-12, 1.731e-12,
02681 1.981e-12, 2.387e-12, 2.93e-12, 3.642e-12, 4.584e-12, 5.822e-12,
02682 7.278e-12, 9.193e-12, 1.135e-11, 1.382e-11, 1.662e-11, 1.958e-11,
02683 2.286e-11, 2.559e-11, 2.805e-11, 2.988e-11, 3.106e-11, 3.182e-11,
02684 3.2e-11, 3.258e-11, 3.362e-11, 3.558e-11, 3.688e-11, 3.8e-11,
02685 3.929e-11, 4.062e-11, 4.186e-11, 4.293e-11, 4.48e-11, 4.643e-11,
02686 4.704e-11, 4.571e-11, 4.206e-11, 3.715e-11, 3.131e-11, 2.541e-11,
02687 1.978e-11, 1.508e-11, 1.146e-11, 8.7e-12, 6.603e-12, 5.162e-12,
02688 4.157e-12, 3.408e-12, 2.829e-12, 2.405e-12, 2.071e-12, 1.826e-12,
02689 1.648e-12, 1.542e-12, 1.489e-12, 1.485e-12, 1.493e-12, 1.545e-12,
02690 1.637e-12, 1.814e-12, 2.061e-12, 2.312e-12, 2.651e-12, 3.03e-12,
02691 3.46e-12, 3.901e-12, 4.306e-12, 4.721e-12, 5.008e-12, 5.281e-12,
02692 5.541e-12, 5.791e-12, 6.115e-12, 6.442e-12, 6.68e-12, 6.791e-12,
02693 6.831e-12, 6.839e-12, 6.946e-12, 7.128e-12, 7.537e-12, 8.036e-12,
02694 8.392e-12, 8.526e-12, 8.11e-12, 7.325e-12, 6.329e-12, 5.183e-12,
02695 4.081e-12, 2.985e-12, 2.141e-12, 1.492e-12, 1.015e-12, 6.684e-13,
02696 4.414e-13, 2.987e-13, 2.038e-13, 1.391e-13, 9.86e-14, 7.24e-14,
02697 5.493e-14, 4.288e-14, 3.427e-14, 2.787e-14, 2.296e-14, 1.909e-14,
02698 1.598e-14, 1.344e-14, 1.135e-14, 9.616e-15, 8.169e-15, 6.957e-15,
02699 5.938e-15, 5.08e-15, 4.353e-15, 3.738e-15, 3.217e-15, 2.773e-15,
02700 2.397e-15, 2.077e-15, 1.805e-15, 1.575e-15, 1.382e-15, 1.221e-15,
02701 1.09e-15, 9.855e-16, 9.068e-16, 8.537e-16, 8.27e-16, 8.29e-16,
02702 8.634e-16, 9.359e-16, 1.055e-15, 1.233e-15, 1.486e-15, 1.839e-15,
02703 2.326e-15, 2.998e-15, 3.934e-15, 5.256e-15, 7.164e-15, 9.984e-15,
02704 1.427e-14, 2.099e-14, 3.196e-14, 5.121e-14, 7.908e-14, 1.131e-13,
02705 1.602e-13, 2.239e-13, 3.075e-13, 4.134e-13, 5.749e-13, 7.886e-13,
02706 1.071e-12, 1.464e-12, 2.032e-12, 2.8e-12, 3.732e-12, 4.996e-12,
02707 6.483e-12, 8.143e-12, 1.006e-11, 1.238e-11, 1.484e-11, 1.744e-11,
02708 2.02e-11, 2.274e-11, 2.562e-11, 2.848e-11, 3.191e-11, 3.617e-11,
02709 4.081e-11, 4.577e-11, 4.937e-11, 5.204e-11, 5.401e-11, 5.462e-11,
02710 5.507e-11, 5.51e-11, 5.605e-11, 5.686e-11, 5.739e-11, 5.766e-11,
02711 5.74e-11, 5.754e-11, 5.761e-11, 5.777e-11, 5.712e-11, 5.51e-11,
02712 5.088e-11, 4.438e-11, 3.728e-11, 2.994e-11, 2.305e-11, 1.715e-11,
02713 1.256e-11, 9.208e-12, 6.745e-12, 5.014e-12, 3.785e-12, 2.9e-12,
02714 2.239e-12, 1.757e-12, 1.414e-12, 1.142e-12, 9.482e-13, 8.01e-13,
02715 6.961e-13, 6.253e-13, 5.735e-13, 5.433e-13, 5.352e-13, 5.493e-13,
02716 5.706e-13, 6.068e-13, 6.531e-13, 7.109e-13, 7.767e-13, 8.59e-13,
02717 9.792e-13, 1.142e-12, 1.371e-12, 1.65e-12, 1.957e-12, 2.302e-12,
02718 2.705e-12, 3.145e-12, 3.608e-12, 4.071e-12, 4.602e-12, 5.133e-12,
02719 5.572e-12, 5.987e-12, 6.248e-12, 6.533e-12, 6.757e-12, 6.935e-12,
02720 7.224e-12, 7.422e-12, 7.538e-12, 7.547e-12, 7.495e-12, 7.543e-12,
02721 7.725e-12, 8.139e-12, 8.627e-12, 9.146e-12, 9.443e-12, 9.318e-12,
02722 8.649e-12, 7.512e-12, 6.261e-12, 4.915e-12, 3.647e-12, 2.597e-12,
02723 1.785e-12, 1.242e-12, 8.66e-13, 6.207e-13, 4.61e-13, 3.444e-13,
02724 2.634e-13, 2.1e-13, 1.725e-13, 1.455e-13, 1.237e-13, 1.085e-13,
02725 9.513e-14, 7.978e-14, 6.603e-14, 5.288e-14, 4.084e-14, 2.952e-14,
02726 2.157e-14, 1.593e-14, 1.199e-14, 9.267e-15, 7.365e-15, 6.004e-15,
02727 4.995e-15, 4.218e-15, 3.601e-15, 3.101e-15, 2.692e-15, 2.36e-15,
02728 2.094e-15, 1.891e-15, 1.755e-15, 1.699e-15, 1.755e-15, 1.987e-15,
02729 2.506e-15, 3.506e-15, 5.289e-15, 8.311e-15, 1.325e-14, 2.129e-14,
02730 3.237e-14, 4.595e-14, 6.441e-14, 8.433e-14, 1.074e-13, 1.383e-13,
02731 1.762e-13, 2.281e-13, 2.831e-13, 3.523e-13, 4.38e-13, 5.304e-13,
02732 6.29e-13, 7.142e-13, 8.032e-13, 8.934e-13, 9.888e-13, 1.109e-12,
02733 1.261e-12, 1.462e-12, 1.74e-12, 2.099e-12, 2.535e-12, 3.008e-12,
02734 3.462e-12, 3.856e-12, 4.098e-12, 4.239e-12, 4.234e-12, 4.132e-12,
02735 3.986e-12, 3.866e-12, 3.829e-12, 3.742e-12, 3.705e-12, 3.694e-12,
02736 3.765e-12, 3.849e-12, 3.929e-12, 4.056e-12, 4.092e-12, 4.047e-12,
02737 3.792e-12, 3.407e-12, 2.953e-12, 2.429e-12, 1.931e-12, 1.46e-12,
02738 1.099e-12, 8.199e-13, 6.077e-13, 4.449e-13, 3.359e-13, 2.524e-13,
02739 1.881e-13, 1.391e-13, 1.02e-13, 7.544e-14, 5.555e-14, 4.22e-14,
02740 3.321e-14, 2.686e-14, 2.212e-14, 1.78e-14, 1.369e-14, 1.094e-14,
02741 9.13e-15, 8.101e-15, 7.828e-15, 8.393e-15, 1.012e-14, 1.259e-14,
02742 1.538e-14, 1.961e-14, 2.619e-14, 3.679e-14, 5.049e-14, 6.917e-14,
02743 8.88e-14, 1.115e-13, 1.373e-13, 1.619e-13, 1.878e-13, 2.111e-13,
02744 2.33e-13, 2.503e-13, 2.613e-13, 2.743e-13, 2.826e-13, 2.976e-13,
02745 3.162e-13, 3.36e-13, 3.491e-13, 3.541e-13, 3.595e-13, 3.608e-13,
02746 3.709e-13, 3.869e-13, 4.12e-13, 4.366e-13, 4.504e-13, 4.379e-13,
02747 3.955e-13, 3.385e-13, 2.741e-13, 2.089e-13, 1.427e-13, 9.294e-14,
02748 5.775e-14, 3.565e-14, 2.21e-14, 1.398e-14, 9.194e-15, 6.363e-15,
02749 4.644e-15, 3.55e-15, 2.808e-15, 2.274e-15, 1.871e-15, 1.557e-15,
02750 1.308e-15, 1.108e-15, 9.488e-16, 8.222e-16, 7.238e-16, 6.506e-16,
02751 6.008e-16, 5.742e-16, 5.724e-16, 5.991e-16, 6.625e-16, 7.775e-16,
02752 9.734e-16, 1.306e-15, 1.88e-15, 2.879e-15, 4.616e-15, 7.579e-15,
02753 1.248e-14, 2.03e-14, 3.244e-14, 5.171e-14, 7.394e-14, 9.676e-14,
02754 1.199e-13, 1.467e-13, 1.737e-13, 2.02e-13, 2.425e-13, 3.016e-13,
02755 3.7e-13, 4.617e-13, 5.949e-13, 7.473e-13, 9.378e-13, 1.191e-12,
02756 1.481e-12, 1.813e-12, 2.232e-12, 2.722e-12, 3.254e-12, 3.845e-12,
02757 4.458e-12, 5.048e-12, 5.511e-12, 5.898e-12, 6.204e-12, 6.293e-12,
02758 6.386e-12, 6.467e-12, 6.507e-12, 6.466e-12, 6.443e-12, 6.598e-12,
02759 6.873e-12, 7.3e-12, 7.816e-12, 8.368e-12, 8.643e-12, 8.466e-12,
```

```

02760      7.871e-12, 6.853e-12, 5.714e-12, 4.482e-12, 3.392e-12, 2.613e-12,
02761      2.008e-12, 1.562e-12, 1.228e-12, 9.888e-13, 7.646e-13, 5.769e-13,
02762      4.368e-13, 3.324e-13, 2.508e-13, 1.916e-13
02763  };
02764
02765  static double xfcrev[15] =
02766  { 1.003, 1.009, 1.015, 1.023, 1.029, 1.033, 1.037,
02767    1.039, 1.04, 1.046, 1.036, 1.027, 1.01, 1.002, 1.
02768  };
02769
02770  double sfac;
02771
02772  /* Get H2O continuum absorption... */
02773  double xw = nu / 10 + 1;
02774  if (xw >= 1 && xw < 2001) {
02775      int iw = (int) xw;
02776      double dw = xw - iw;
02777      double ew = 1 - dw;
02778      double cw296 = ew * h2o296[iw - 1] + dw * h2o296[iw];
02779      double cw260 = ew * h2o260[iw - 1] + dw * h2o260[iw];
02780      double cwfrn = ew * h2ofrn[iw - 1] + dw * h2ofrn[iw];
02781      if (nu <= 820 || nu >= 960) {
02782          sfac = 1;
02783      } else {
02784          double xx = (nu - 820) / 10;
02785          int ix = (int) xx;
02786          double dx = xx - ix;
02787          sfac = (1 - dx) * xfcrev[ix] + dx * xfcrev[ix + 1];
02788      }
02789      double ctws1f =
02790          sfac * cw296 * pow(cw260 / cw296, (296 - t) / (296 - 260));
02791      double vf2 = POW2(nu - 370);
02792      double vf6 = POW3(vf2);
02793      double fscal = 36100 / (vf2 + vf6 * 1e-8 + 36100) * -.25 + 1;
02794      double ctwfrn = cwfrn * fscal;
02795      double a1 = nu * u * tanh(.7193876 / t * nu);
02796      double a2 = 296 / t;
02797      double a3 = p / P0 * (q * ctws1f + (1 - q) * ctwfrn) * 1e-20;
02798      return a1 * a2 * a3;
02799  } else
02800      return 0;
02801 }

```

**5.19.3.8 ctmn2()** double ctmn2 (  
     double nu,  
     double p,  
     double t )

Compute nitrogen continuum (absorption coefficient).

Definition at line 2805 of file [jurassic.c](#).

```

02808      {
02809
02810      static double ba[98] = { 0., 4.45e-8, 5.22e-8, 6.46e-8, 7.75e-8, 9.03e-8,
02811      1.06e-7, 1.21e-7, 1.37e-7, 1.57e-7, 1.75e-7, 2.01e-7, 2.3e-7,
02812      2.59e-7, 2.95e-7, 3.26e-7, 3.66e-7, 4.05e-7, 4.47e-7, 4.92e-7,
02813      5.34e-7, 5.84e-7, 6.24e-7, 6.67e-7, 7.14e-7, 7.26e-7, 7.54e-7,
02814      7.84e-7, 8.09e-7, 8.42e-7, 8.62e-7, 8.87e-7, 9.11e-7, 9.36e-7,
02815      9.76e-7, 1.03e-6, 1.11e-6, 1.23e-6, 1.39e-6, 1.61e-6, 1.76e-6,
02816      1.94e-6, 1.97e-6, 1.87e-6, 1.75e-6, 1.56e-6, 1.42e-6, 1.35e-6,
02817      1.32e-6, 1.29e-6, 1.29e-6, 1.29e-6, 1.3e-6, 1.32e-6, 1.33e-6,
02818      1.34e-6, 1.35e-6, 1.33e-6, 1.31e-6, 1.29e-6, 1.24e-6, 1.2e-6,
02819      1.16e-6, 1.1e-6, 1.04e-6, 9.96e-7, 9.38e-7, 8.63e-7, 7.98e-7,
02820      7.26e-7, 6.55e-7, 5.94e-7, 5.35e-7, 4.74e-7, 4.24e-7, 3.77e-7,
02821      3.33e-7, 2.96e-7, 2.63e-7, 2.34e-7, 2.08e-7, 1.85e-7, 1.67e-7,
02822      1.47e-7, 1.32e-7, 1.2e-7, 1.09e-7, 9.85e-8, 9.08e-8, 8.18e-8,
02823      7.56e-8, 6.85e-8, 6.14e-8, 5.83e-8, 5.77e-8, 5e-8, 4.32e-8, 0.
02824  };
02825
02826  static double betaa[98] = { 802., 802., 761., 722., 679., 646., 609., 562.,
02827  511., 472., 436., 406., 377., 355., 338., 319., 299., 278., 255.,
02828  233., 208., 184., 149., 107., 66., 25., -13., -49., -82., -104.,
02829  -119., -130., -139., -144., -146., -146., -147., -148., -150.,
02830  -153., -160., -169., -181., -189., -195., -200., -205., -209.,
02831  -211., -210., -210., -209., -205., -199., -190., -180., -168.,
02832  -157., -143., -126., -108., -89., -63., -32., 1., 35., 65., 95.,
02833  121., 141., 152., 161., 164., 164., 161., 155., 148., 143., 137.,
02834  133., 131., 133., 139., 150., 165., 187., 213., 248., 284., 321.,

```

```

02835     372., 449., 514., 569., 609., 642., 673., 673.
02836 };
02837
02838 static double nua[98] = { 2120., 2125., 2130., 2135., 2140., 2145., 2150.,
02839     2155., 2160., 2165., 2170., 2175., 2180., 2185., 2190., 2195.,
02840     2200., 2205., 2210., 2215., 2220., 2225., 2230., 2235., 2240.,
02841     2245., 2250., 2255., 2260., 2265., 2270., 2275., 2280., 2285.,
02842     2290., 2295., 2300., 2305., 2310., 2315., 2320., 2325., 2330.,
02843     2335., 2340., 2345., 2350., 2355., 2360., 2365., 2370., 2375.,
02844     2380., 2385., 2390., 2395., 2400., 2405., 2410., 2415., 2420.,
02845     2425., 2430., 2435., 2440., 2445., 2450., 2455., 2460., 2465.,
02846     2470., 2475., 2480., 2485., 2490., 2495., 2500., 2505., 2510.,
02847     2515., 2520., 2525., 2530., 2535., 2540., 2545., 2550., 2555.,
02848     2560., 2565., 2570., 2575., 2580., 2585., 2590., 2595., 2600., 2605.
02849 };
02850
02851 const double q_n2 = 0.79, t0 = 273.0, tr = 296.0;
02852
02853 /* Check wavenumber range... */
02854 if (nu < nua[0] || nu > nua[97])
02855     return 0;
02856
02857 /* Interpolate B and beta... */
02858 int idx = locate_reg(nua, 98, nu);
02859 double b = LIN(nua[idx], ba[idx], nua[idx + 1], ba[idx + 1], nu);
02860 double beta = LIN(nua[idx], betaa[idx], nua[idx + 1], betaa[idx + 1], nu);
02861
02862 /* Compute absorption coefficient... */
02863 return 0.1 * POW2(p / P0 * t0 / t) * exp(beta * (1 / tr - 1 / t))
02864     * q_n2 * b * (q_n2 + (1 - q_n2) * (1.294 - 0.4545 * t / tr));
02865 }

```

Here is the call graph for this function:



### 5.19.3.9 ctmo2()

```

double ctmo2 (
    double nu,
    double p,
    double t )

```

Compute oxygen continuum (absorption coefficient).

Definition at line 2869 of file [jurassic.c](#).

```

02872     {
02873
02874 static double ba[90] = { 0., .061, .074, .084, .096, .12, .162, .208, .246,
02875     .285, .314, .38, .444, .5, .571, .673, .768, .853, .966, 1.097,
02876     1.214, 1.333, 1.466, 1.591, 1.693, 1.796, 1.922, 2.037, 2.154,
02877     2.264, 2.375, 2.508, 2.671, 2.847, 3.066, 3.417, 3.828, 4.204,
02878     4.453, 4.599, 4.528, 4.284, 3.955, 3.678, 3.477, 3.346, 3.29,
02879     3.251, 3.231, 3.226, 3.212, 3.192, 3.108, 3.033, 2.911, 2.798,
02880     2.646, 2.508, 2.322, 2.13, 1.928, 1.757, 1.588, 1.417, 1.253,
02881     1.109, .99, .888, .791, .678, .587, .524, .464, .403, .357, .32,
02882     .29, .267, .242, .215, .182, .16, .146, .128, .103, .087, .081,
02883     .071, .064, 0.
02884 };
02885
02886 static double betaa[90] = { 467., 467., 400., 315., 379., 368., 475., 521.,
02887     531., 512., 442., 444., 430., 381., 335., 324., 296., 248., 215.,
02888     193., 158., 127., 101., 71., 31., -6., -26., -47., -63., -79.,
02889     -88., -88., -87., -90., -98., -99., -109., -134., -160., -167.,

```

```

02890     -164., -158., -153., -151., -156., -166., -168., -173., -170.,
02891     -161., -145., -126., -108., -84., -59., -29., 4., 41., 73., 97.,
02892     123., 159., 198., 220., 242., 256., 281., 311., 334., 319., 313.,
02893     321., 323., 310., 315., 320., 335., 361., 378., 373., 338., 319.,
02894     346., 322., 291., 290., 350., 371., 504., 504.
02895 };
02896
02897 static double nua[90] = { 1360., 1365., 1370., 1375., 1380., 1385., 1390.,
02898     1395., 1400., 1405., 1410., 1415., 1420., 1425., 1430., 1435.,
02899     1440., 1445., 1450., 1455., 1460., 1465., 1470., 1475., 1480.,
02900     1485., 1490., 1495., 1500., 1505., 1510., 1515., 1520., 1525.,
02901     1530., 1535., 1540., 1545., 1550., 1555., 1560., 1565., 1570.,
02902     1575., 1580., 1585., 1590., 1595., 1600., 1605., 1610., 1615.,
02903     1620., 1625., 1630., 1635., 1640., 1645., 1650., 1655., 1660.,
02904     1665., 1670., 1675., 1680., 1685., 1690., 1695., 1700., 1705.,
02905     1710., 1715., 1720., 1725., 1730., 1735., 1740., 1745., 1750.,
02906     1755., 1760., 1765., 1770., 1775., 1780., 1785., 1790., 1795.,
02907     1800., 1805.
02908 };
02909
02910 const double q_o2 = 0.21, t0 = 273, tr = 296;
02911
02912 /* Check wavenumber range... */
02913 if (nu < nua[0] || nu > nua[89])
02914     return 0;
02915
02916 /* Interpolate B and beta... */
02917 int idx = locate_reg(nua, 90, nu);
02918 double b = LIN(nua[idx], ba[idx], nua[idx + 1], ba[idx + 1], nu);
02919 double beta = LIN(nua[idx], betaa[idx], nua[idx + 1], betaa[idx + 1], nu);
02920
02921 /* Compute absorption coefficient... */
02922 return 0.1 * POW2(p / P0 * t0 / t) * exp(beta * (1 / tr - 1 / t)) * q_o2 *
02923     b;
02924 }

```

Here is the call graph for this function:



**5.19.3.10 copy\_atm()** void copy\_atm (

```

    ctl_t * ctl,
    atm_t * atm_dest,
    atm_t * atm_src,
    int init )

```

Copy and initialize atmospheric data.

Definition at line 2928 of file [jurassic.c](#).

```

02932     {
02933
02934     /* Data size... */
02935     size_t s = (size_t) atm_src->np * sizeof(double);
02936
02937     /* Copy data... */
02938     atm_dest->np = atm_src->np;
02939     memcpy(atm_dest->time, atm_src->time, s);
02940     memcpy(atm_dest->z, atm_src->z, s);
02941     memcpy(atm_dest->lon, atm_src->lon, s);
02942     memcpy(atm_dest->lat, atm_src->lat, s);
02943     memcpy(atm_dest->p, atm_src->p, s);
02944     memcpy(atm_dest->t, atm_src->t, s);

```

```

02945     for (int ig = 0; ig < ctl->ng; ig++)
02946         memcpy(atm_dest->q[ig], atm_src->q[ig], s);
02947     for (int iw = 0; iw < ctl->nw; iw++)
02948         memcpy(atm_dest->k[iw], atm_src->k[iw], s);
02949     atm_dest->clz = atm_src->clz;
02950     atm_dest->cldz = atm_src->cldz;
02951     for (int icl = 0; icl < ctl->ncl; icl++)
02952         atm_dest->clk[icl] = atm_src->clk[icl];
02953     atm_dest->sfz = atm_src->sfz;
02954     atm_dest->sfp = atm_src->sfp;
02955     atm_dest->sft = atm_src->sft;
02956     for (int isf = 0; isf < ctl->nsf; isf++)
02957         atm_dest->sfeps[isf] = atm_src->sfeps[isf];
02958
02959     /* Initialize... */
02960     if (init)
02961         for (int ip = 0; ip < atm_dest->np; ip++) {
02962             atm_dest->p[ip] = 0;
02963             atm_dest->t[ip] = 0;
02964             for (int ig = 0; ig < ctl->ng; ig++)
02965                 atm_dest->q[ig][ip] = 0;
02966             for (int iw = 0; iw < ctl->nw; iw++)
02967                 atm_dest->k[iw][ip] = 0;
02968             atm_dest->clz = 0;
02969             atm_dest->cldz = 0;
02970             for (int icl = 0; icl < ctl->ncl; icl++)
02971                 atm_dest->clk[icl] = 0;
02972             atm_dest->sfz = 0;
02973             atm_dest->sfp = 0;
02974             atm_dest->sft = 0;
02975             for (int isf = 0; isf < ctl->nsf; isf++)
02976                 atm_dest->sfeps[isf] = 1;
02977         }
02978 }

```

**5.19.3.11 copy\_obs()** void copy\_obs (

```

    ctl_t * ctl,
    obs_t * obs_dest,
    obs_t * obs_src,
    int init )

```

Copy and initialize observation data.

Definition at line 2982 of file [jurassic.c](#).

```

02986     {
02987
02988         /* Data size... */
02989         size_t s = (size_t) obs_src->nr * sizeof(double);
02990
02991         /* Copy data... */
02992         obs_dest->nr = obs_src->nr;
02993         memcpy(obs_dest->time, obs_src->time, s);
02994         memcpy(obs_dest->obsz, obs_src->obsz, s);
02995         memcpy(obs_dest->obslon, obs_src->obslon, s);
02996         memcpy(obs_dest->obslat, obs_src->obslat, s);
02997         memcpy(obs_dest->vpz, obs_src->vpz, s);
02998         memcpy(obs_dest->vplon, obs_src->vplon, s);
02999         memcpy(obs_dest->vplat, obs_src->vplat, s);
03000         memcpy(obs_dest->tpz, obs_src->tpz, s);
03001         memcpy(obs_dest->tplon, obs_src->tplon, s);
03002         memcpy(obs_dest->tplat, obs_src->tplat, s);
03003         for (int id = 0; id < ctl->nd; id++)
03004             memcpy(obs_dest->rad[id], obs_src->rad[id], s);
03005         for (int id = 0; id < ctl->nd; id++)
03006             memcpy(obs_dest->tau[id], obs_src->tau[id], s);
03007
03008         /* Initialize... */
03009         if (init)
03010             for (int id = 0; id < ctl->nd; id++)
03011                 for (int ir = 0; ir < obs_dest->nr; ir++)
03012                     if (gsl_finite(obs_dest->rad[id][ir])) {
03013                         obs_dest->rad[id][ir] = 0;
03014                         obs_dest->tau[id][ir] = 0;
03015                     }
03016 }

```



**5.19.3.12 find\_emitter()** int find\_emitter (  
     ctl\_t \* ctl,  
     const char \* emitter )

Find index of an emitter.

Definition at line 3020 of file [jurassic.c](#).

```
03022     {
03023
03024         for (int ig = 0; ig < ctl->ng; ig++)
03025             if (strcasecmp(ctl->emitter[ig], emitter) == 0)
03026                 return ig;
03027
03028         return -1;
03029     }
```

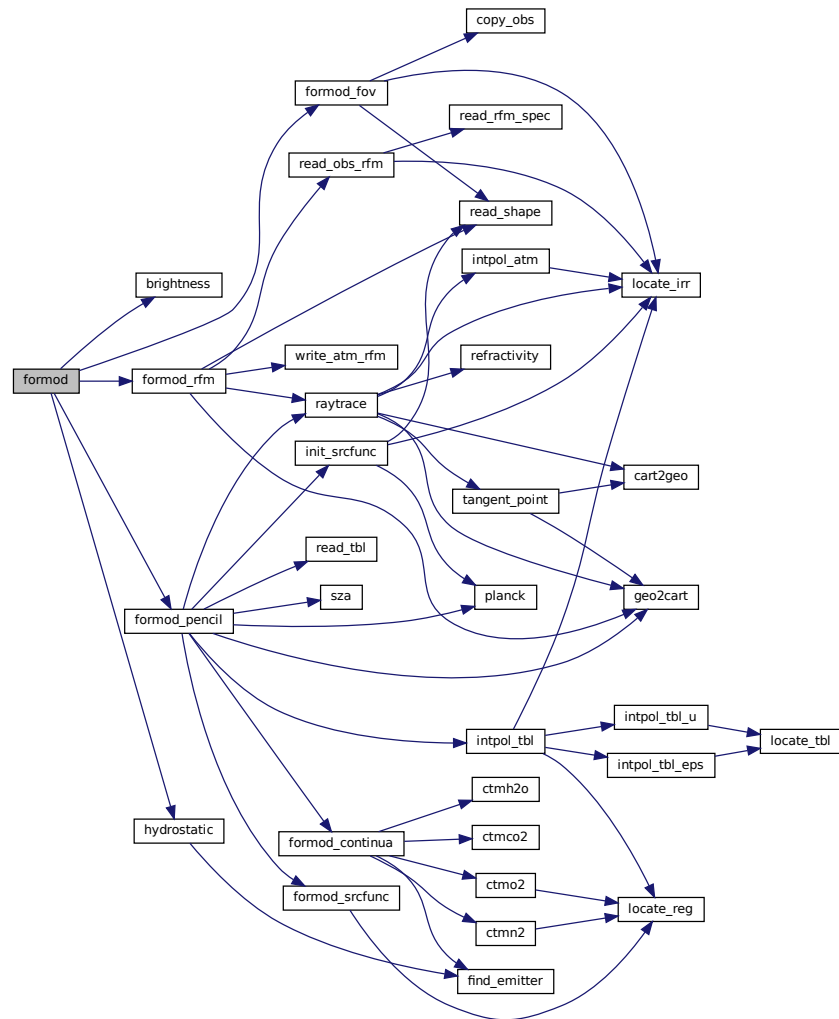
**5.19.3.13 formod()** void formod (  
     ctl\_t \* ctl,  
     atm\_t \* atm,  
     obs\_t \* obs )

Determine ray paths and compute radiative transfer.

Definition at line 3033 of file [jurassic.c](#).

```
03036     {
03037
03038         int *mask;
03039
03040         /* Allocate... */
03041         ALLOC(mask, int,
03042               ND * NR);
03043
03044         /* Save observation mask... */
03045         for (int id = 0; id < ctl->nd; id++)
03046             for (int ir = 0; ir < obs->nr; ir++)
03047                 mask[id * NR + ir] = !gsl_finite(obs->rad[id][ir]);
03048
03049         /* Hydrostatic equilibrium... */
03050         hydrostatic(ctl, atm);
03051
03052         /* EGA forward model... */
03053         if (ctl->formod == 1)
03054             for (int ir = 0; ir < obs->nr; ir++)
03055                 formod_pencil(ctl, atm, obs, ir);
03056
03057         /* Call RFM... */
03058         else if (ctl->formod == 2)
03059             formod_rfm(ctl, atm, obs);
03060
03061         /* Apply field-of-view convolution... */
03062         formod_fov(ctl, obs);
03063
03064         /* Convert radiance to brightness temperature... */
03065         if (ctl->write_bbt)
03066             for (int id = 0; id < ctl->nd; id++)
03067                 for (int ir = 0; ir < obs->nr; ir++)
03068                     obs->rad[id][ir] = brightness(obs->rad[id][ir], ctl->nu[id]);
03069
03070         /* Apply observation mask... */
03071         for (int id = 0; id < ctl->nd; id++)
03072             for (int ir = 0; ir < obs->nr; ir++)
03073                 if (mask[id * NR + ir])
03074                     obs->rad[id][ir] = GSL_NAN;
03075
03076         /* Free... */
03077         free(mask);
03078     }
```

Here is the call graph for this function:



**5.19.3.14 formod\_continua()** void formod\_continua (

```

    ctl_t * ctl,
    los_t * los,
    int ip,
    double * beta )

```

Compute absorption coefficient of continua.

Definition at line 3082 of file [jurassic.c](#).

```

03086     {
03087
03088     static int ig_co2 = -999, ig_h2o = -999;
03089
03090     /* Extinction... */
03091     for (int id = 0; id < ctl->nd; id++)
03092         beta[id] = los->k[ip][id];
03093
03094     /* CO2 continuum... */
03095     if (ctl->ctm_co2) {

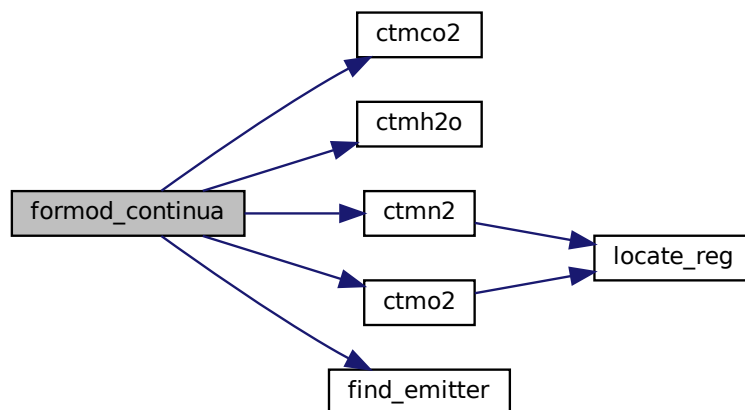
```

```

03096     if (ig_co2 == -999)
03097         ig_co2 = find_emitter(ctl, "CO2");
03098     if (ig_co2 >= 0)
03099         for (int id = 0; id < ctl->nd; id++)
03100             beta[id] += ctmc2(ctl->nu[id], los->p[ip], los->t[ip],
03101                             los->u[ip][ig_co2]) / los->ds[ip];
03102     }
03103
03104     /* H2O continuum... */
03105     if (ctl->ctm_h2o) {
03106         if (ig_h2o == -999)
03107             ig_h2o = find_emitter(ctl, "H2O");
03108         if (ig_h2o >= 0)
03109             for (int id = 0; id < ctl->nd; id++)
03110                 beta[id] += ctmh2o(ctl->nu[id], los->p[ip], los->t[ip],
03111                                    los->q[ip][ig_h2o], los->u[ip][ig_h2o])
03112                                / los->ds[ip];
03113     }
03114
03115     /* N2 continuum... */
03116     if (ctl->ctm_n2)
03117         for (int id = 0; id < ctl->nd; id++)
03118             beta[id] += ctmn2(ctl->nu[id], los->p[ip], los->t[ip]);
03119
03120     /* O2 continuum... */
03121     if (ctl->ctm_o2)
03122         for (int id = 0; id < ctl->nd; id++)
03123             beta[id] += ctmo2(ctl->nu[id], los->p[ip], los->t[ip]);
03124 }

```

Here is the call graph for this function:



**5.19.3.15 formod\_fov()** void formod\_fov (  
     ctl\_t \* ctl,  
     obs\_t \* obs )

Apply field of view convolution.

Definition at line 3128 of file [jurassic.c](#).

```

03130     {
03131
03132         static double dz[NSHAPE], w[NSHAPE];
03133
03134         static int init = 0, n;
03135

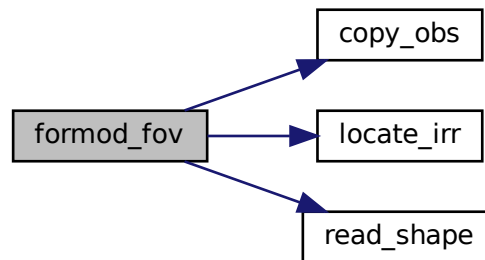
```

```

03136     obs_t *obs2;
03137
03138     double rad[ND][NR], tau[ND][NR], z[NR];
03139
03140     /* Do not take into account FOV... */
03141     if (ctl->fov[0] == '-')
03142         return;
03143
03144     /* Initialize FOV data... */
03145     if (!init) {
03146         init = 1;
03147         read_shape(ctl->fov, dz, w, &n);
03148     }
03149
03150     /* Allocate... */
03151     ALLOC(obs2, obs_t, 1);
03152
03153     /* Copy observation data... */
03154     copy_obs(ctl, obs2, obs, 0);
03155
03156     /* Loop over ray paths... */
03157     for (int ir = 0; ir < obs->nr; ir++) {
03158
03159         /* Get radiance and transmittance profiles... */
03160         int nz = 0;
03161         for (int ir2 = GSL_MAX(ir - NFOV, 0);
03162              ir2 < GSL_MIN(ir + 1 + NFOV, obs->nr); ir2++)
03163             if (obs->time[ir2] == obs->time[ir]) {
03164                 z[nz] = obs2->vpz[ir2];
03165                 for (int id = 0; id < ctl->nd; id++) {
03166                     rad[id][nz] = obs2->rad[id][ir2];
03167                     tau[id][nz] = obs2->tau[id][ir2];
03168                 }
03169                 nz++;
03170             }
03171         if (nz < 2)
03172             ERRMSG("Cannot apply FOV convolution!");
03173
03174         /* Convolute profiles with FOV... */
03175         double wsum = 0;
03176         for (int id = 0; id < ctl->nd; id++) {
03177             obs->rad[id][ir] = 0;
03178             obs->tau[id][ir] = 0;
03179         }
03180         for (int i = 0; i < n; i++) {
03181             double zfov = obs->vpz[ir] + dz[i];
03182             int idx = locate_irr(z, nz, zfov);
03183             for (int id = 0; id < ctl->nd; id++) {
03184                 obs->rad[id][ir] += w[i]
03185                     * LIN(z[idx], rad[id][idx], z[idx + 1], rad[id][idx + 1], zfov);
03186                 obs->tau[id][ir] += w[i]
03187                     * LIN(z[idx], tau[id][idx], z[idx + 1], tau[id][idx + 1], zfov);
03188             }
03189             wsum += w[i];
03190         }
03191         for (int id = 0; id < ctl->nd; id++) {
03192             obs->rad[id][ir] /= wsum;
03193             obs->tau[id][ir] /= wsum;
03194         }
03195     }
03196
03197     /* Free... */
03198     free(obs2);
03199 }

```

Here is the call graph for this function:



**5.19.3.16 formod\_pencil()** void formod\_pencil (  
     ctl\_t \* ctl,  
     atm\_t \* atm,  
     obs\_t \* obs,  
     int ir )

Compute radiative transfer for a pencil beam.

Definition at line 3203 of file jurassic.c.

```

03207     {
03208
03209     static tbl_t *tbl;
03210
03211     static int init = 0;
03212
03213     los_t *los;
03214
03215     double beta_ctm[ND], rad[ND], tau[ND], tau_refl[ND],
03216           tau_path[ND][NG], tau_gas[ND], x0[3], x1[3];
03217
03218     /* Initialize look-up tables... */
03219     if (!init) {
03220         init = 1;
03221         ALLOC(tbl, tbl_t, 1);
03222         read_tbl(ctl, tbl);
03223         init_srcfunc(ctl, tbl);
03224     }
03225
03226     /* Allocate... */
03227     ALLOC(los, los_t, 1);
03228
03229     /* Initialize... */
03230     for (int id = 0; id < ctl->nd; id++) {
03231         rad[id] = 0;
03232         tau[id] = 1;
03233         for (int ig = 0; ig < ctl->ng; ig++)
03234             tau_path[id][ig] = 1;
03235     }
03236
03237     /* Raytracing... */
03238     raytrace(ctl, atm, obs, los, ir);
03239
03240     /* Loop over LOS points... */
03241     for (int ip = 0; ip < los->np; ip++) {
03242
03243         /* Get trace gas transmittance... */
03244         intpol_tbl(ctl, tbl, los, ip, tau_path, tau_gas);
03245
03246         /* Get continuum absorption... */
  
```

```

03247     formod_continua(ctl, los, ip, beta_ctm);
03248
03249     /* Compute Planck function... */
03250     formod_srcfunc(ctl, tbl, los->t[ip], los->src[ip]);
03251
03252     /* Loop over channels... */
03253     for (int id = 0; id < ctl->nd; id++)
03254     {
03255         /* Get segment emissivity... */
03256         los->eps[ip][id] = 1 - tau_gas[id] * exp(-beta_ctm[id] * los->ds[ip]);
03257
03258         /* Compute radiance... */
03259         rad[id] += los->src[ip][id] * los->eps[ip][id] * tau[id];
03260
03261         /* Compute path transmittance... */
03262         tau[id] *= (1 - los->eps[ip][id]);
03263     }
03264 }
03265
03266 /* Check whether LOS hit the ground... */
03267 if (ctl->sftype >= 1 && los->sft > 0) {
03268
03269     /* Add surface emissions... */
03270     double src_sf[ND];
03271     formod_srcfunc(ctl, tbl, los->sft, src_sf);
03272     for (int id = 0; id < ctl->nd; id++)
03273         rad[id] += los->sfeps[id] * src_sf[id] * tau[id];
03274
03275     /* Check reflectivity... */
03276     int refl = 0;
03277     if (ctl->sftype >= 2)
03278         for (int id = 0; id < ctl->nd; id++)
03279             if (los->sfeps[id] < 1) {
03280                 refl = 1;
03281                 break;
03282             }
03283
03284     /* Calculate reflection... */
03285     if (refl) {
03286         /* Initialize... */
03287         for (int id = 0; id < ctl->nd; id++)
03288             tau_refl[id] = 1;
03289
03290         /* Add down-welling radiance... */
03291         for (int ip = los->np - 1; ip >= 0; ip--)
03292             for (int id = 0; id < ctl->nd; id++) {
03293                 rad[id] += los->src[ip][id] * los->eps[ip][id] * tau_refl[id]
03294                     * tau[id] * (1 - los->sfeps[id]);
03295                 tau_refl[id] *= (1 - los->eps[ip][id]);
03296             }
03297
03298         /* Add solar term... */
03299         if (ctl->sftype >= 3) {
03300             /* Get solar zenith angle... */
03301             double sza2;
03302             if (ctl->sfsza < 0)
03303                 sza2 =
03304                     sza(obs->time[ir], los->lon[los->np - 1], los->lat[los->np - 1]);
03305             else
03306                 sza2 = ctl->sfsza;
03307
03308             /* Check solar zenith angle... */
03309             if (sza2 < 89.999) {
03310
03311                 /* Get angle of incidence... */
03312                 geo2cart(los->z[los->np - 1], los->lon[los->np - 1],
03313                     los->lat[los->np - 1], x0);
03314                 geo2cart(los->z[0], los->lon[0], los->lat[0], x1);
03315                 for (int i = 0; i < 3; i++)
03316                     x1[i] -= x0[i];
03317                 double cosa = DOTP(x0, x1) / NORM(x0) / NORM(x1);
03318
03319                 /* Get ratio of SZA and incident radiation... */
03320                 double rcos = cosa / cos(sza2 * M_PI / 180.);
03321
03322                 /* Add solar radiation... */
03323                 for (int id = 0; id < ctl->nd; id++)
03324                     rad[id] += 6.764e-5 / (2. * M_PI) * planck(TSUN, ctl->nu[id])
03325                         * tau_refl[id] * (1 - los->sfeps[id]) * tau[id] * rcos;
03326             }
03327         }
03328     }
03329 }
03330
03331 }
03332 }
03333

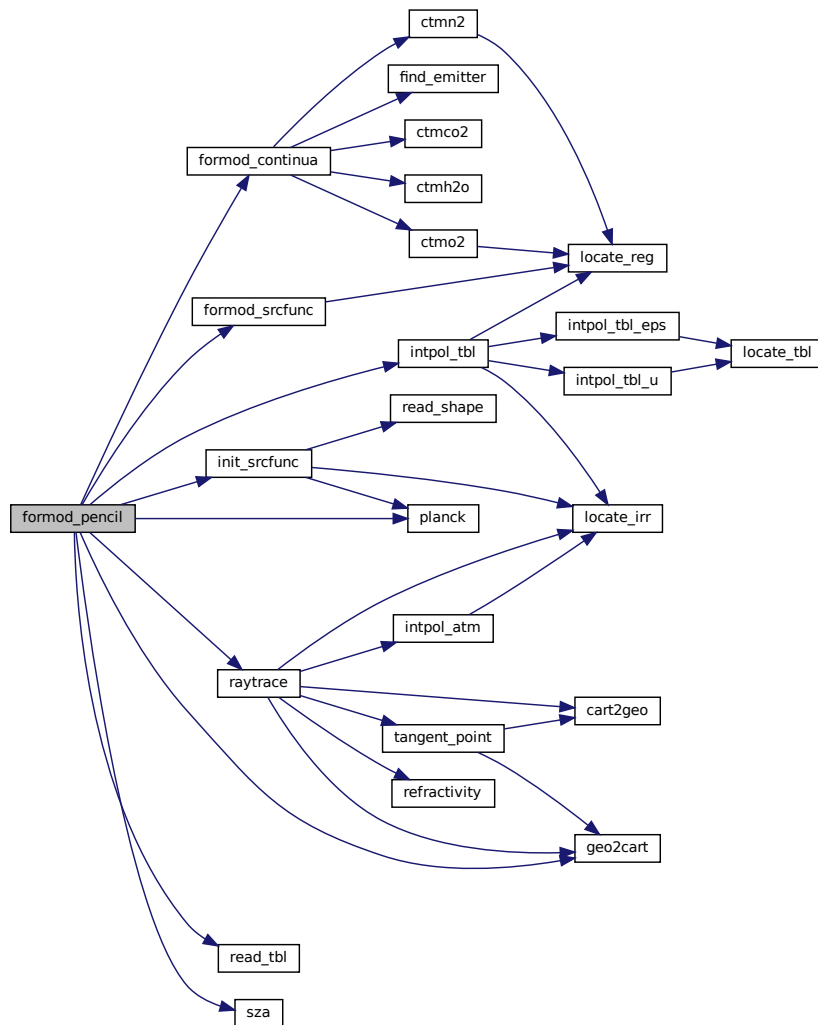
```

```

03334  /* Copy results... */
03335  for (int id = 0; id < ctl->nd; id++) {
03336      obs->rad[id][ir] = rad[id];
03337      obs->tau[id][ir] = tau[id];
03338  }
03339
03340  /* Free... */
03341  free(los);
03342  }

```

Here is the call graph for this function:



**5.19.3.17 formod\_rfm()** void formod\_rfm (  
     ctl\_t \* ctl,  
     atm\_t \* atm,  
     obs\_t \* obs )

Apply RFM for radiative transfer calculations.

Definition at line 3346 of file [jurassic.c](#).

```

03349         {
03350
03351     los_t *los;
03352
03353     FILE *out;
03354
03355     char cmd[2 * LEN], filename[2 * LEN],
03356         rfmflg[LEN] = { "RAD TRA MIX LIN SFC" };
03357
03358     double f[NSHAPE], nu[NSHAPE], nu0, nu1, obsz = -999, tsurf,
03359         xd[3], xo[3], xv[3], z[NR], zmin, zmax;
03360
03361     int i, id, ig, ip, ir, iw, n, nadir = 0;
03362
03363     /* Allocate... */
03364     ALLOC(los, los_t, 1);
03365
03366     /* Check observer positions... */
03367     for (ir = 1; ir < obs->nr; ir++)
03368         if (obs->obsz[ir] != obs->obsz[0]
03369             || obs->obslon[ir] != obs->obslon[0]
03370             || obs->obslat[ir] != obs->obslat[0])
03371             ERRMSG("RFM interface requires identical observer positions!");
03372
03373     /* Check extinction data... */
03374     for (iw = 0; iw < ctl->nw; iw++)
03375         for (ip = 0; ip < atm->np; ip++)
03376             if (atm->k[iw][ip] != 0)
03377                 ERRMSG("RFM interface cannot handle extinction data!");
03378
03379     /* Get altitude range of atmospheric data... */
03380     gsl_stats_minmax(&zmin, &zmax, atm->z, 1, (size_t) atm->np);
03381
03382     /* Observer within atmosphere? */
03383     if (obs->obsz[0] >= zmin && obs->obsz[0] <= zmax) {
03384         obsz = obs->obsz[0];
03385         strcat(rfmflg, " OBS");
03386     }
03387
03388     /* Determine tangent altitude or air mass factor... */
03389     for (ir = 0; ir < obs->nr; ir++) {
03390
03391         /* Raytracing... */
03392         raytrace(ctl, atm, obs, los, ir);
03393
03394         /* Nadir? */
03395         if (obs->tpz[ir] <= zmin) {
03396             geo2cart(obs->obsz[ir], obs->obslon[ir], obs->obslat[ir], xo);
03397             geo2cart(obs->vpz[ir], obs->vpplon[ir], obs->vpplat[ir], xv);
03398             for (i = 0; i < 3; i++)
03399                 xd[i] = xo[i] - xv[i];
03400             z[ir] = NORM(xo) * NORM(xd) / DOTP(xo, xd);
03401             nadir++;
03402         } else
03403             z[ir] = obs->tpz[ir];
03404     }
03405     if (nadir > 0 && nadir < obs->nr)
03406         ERRMSG("Limb and nadir not simultaneously possible!");
03407
03408     /* Nadir? */
03409     if (nadir)
03410         strcat(rfmflg, " NAD");
03411
03412     /* Get surface temperature... */
03413     tsurf = atm->t[gsl_stats_min_index(atm->z, 1, (size_t) atm->np)];
03414
03415     /* Refraction? */
03416     if (!nadir && !ctl->refrac)
03417         strcat(rfmflg, " GEO");
03418
03419     /* Continua? */
03420     if (ctl->ctm_co2 || ctl->ctm_h2o || ctl->ctm_n2 || ctl->ctm_o2)
03421         strcat(rfmflg, " CTM");
03422
03423     /* Write atmospheric data file... */
03424     write_atm_rfm("rfm.atm", ctl, atm);
03425
03426     /* Loop over channels... */
03427     for (id = 0; id < ctl->nd; id++) {
03428
03429         /* Read filter function... */
03430         sprintf(filename, "%s_%.4f.filt", ctl->tblbase, ctl->nu[id]);
03431         read_shape(filename, nu, f, &n);
03432
03433         /* Set spectral range... */
03434         nu0 = nu[0];
03435         nu1 = nu[n - 1];

```

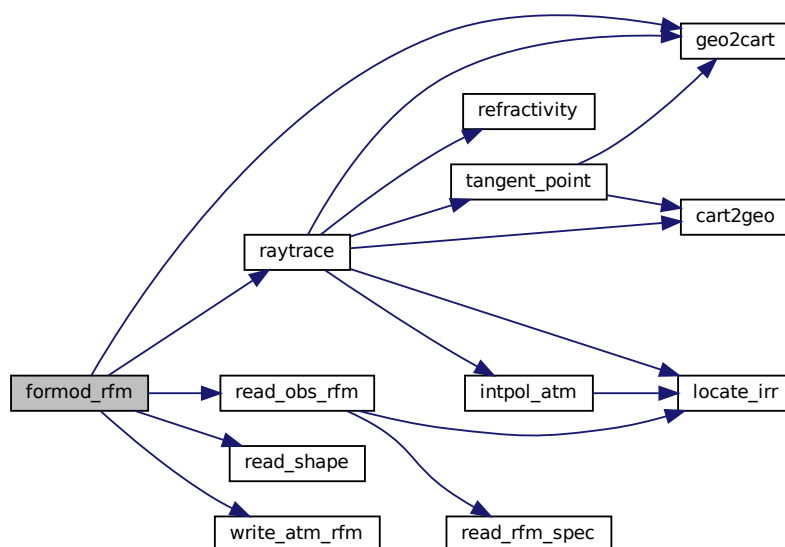


```

03436
03437  /* Create RFM driver file... */
03438  if (!(out = fopen("rfm.drv", "w")))
03439      ERRMSG("Cannot create file!");
03440  fprintf(out, "HDR\nRFM call by JURASSIC.\n");
03441  fprintf(out, "FLG\n%s\n", rfmflg);
03442  fprintf(out, "SPC\n%.4f %.4f 0.0005\n", nu0, nul);
03443  fprintf(out, "GAS\n");
03444  for (ig = 0; ig < ctl->ng; ig++)
03445      fprintf(out, "s\n", ctl->emitter[ig]);
03446  fprintf(out, "ATM\nrfm.atm\n");
03447  fprintf(out, "TAN\n");
03448  for (ir = 0; ir < obs->nr; ir++)
03449      fprintf(out, "g\n", z[ir]);
03450  fprintf(out, "SFC\n%g 1.0\n", tsurf);
03451  if (obsz >= 0)
03452      fprintf(out, "OBS\n%g\n", obsz);
03453  fprintf(out, "HIT\n%s\n", ctl->rfmhit);
03454  fprintf(out, "XSC\n");
03455  for (ig = 0; ig < ctl->ng; ig++)
03456      if (ctl->rfmxsc[ig][0] != '-')
03457          fprintf(out, "s\n", ctl->rfmxsc[ig]);
03458  fprintf(out, "END\n");
03459  fclose(out);
03460
03461  /* Remove temporary files... */
03462  if (system("rm -f rfm.runlog rad_*.asc tra_*.asc"))
03463      ERRMSG("Cannot remove temporary files!");
03464
03465  /* Call RFM... */
03466  sprintf(cmd, "echo | %s", ctl->rfmbin);
03467  if (system(cmd))
03468      ERRMSG("Error while calling RFM!");
03469
03470  /* Read data... */
03471  for (ir = 0; ir < obs->nr; ir++) {
03472      obs->rad[id][ir] = read_obs_rfm("rad", z[ir], nu, f, n) * 1e-5;
03473      obs->tau[id][ir] = read_obs_rfm("tra", z[ir], nu, f, n);
03474  }
03475  }
03476
03477  /* Remove temporary files... */
03478  if (system("rm -f rfm.drv rfm.atm rfm.runlog rad_*.asc tra_*.asc"))
03479      ERRMSG("Error while removing temporary files!");
03480
03481  /* Free... */
03482  free(los);
03483 }

```

Here is the call graph for this function:



**5.19.3.18 formod\_srcfunc()** void formod\_srcfunc (  
    ctl\_t \* ctl,  
    tbl\_t \* tbl,  
    double t,  
    double \* src )

Compute Planck source function.

Definition at line 3487 of file [jurassic.c](#).

```
03491     {  
03492  
03493     /* Determine index in temperature array... */  
03494     int it = locate_reg(tbl->st, TBLNS, t);  
03495  
03496     /* Interpolate Planck function value... */  
03497     for (int id = 0; id < ctl->nd; id++)  
03498         src[id] = LIN(tbl->st[it], tbl->sr[it][id],  
03499                     tbl->st[it + 1], tbl->sr[it + 1][id], t);  
03500 }
```

Here is the call graph for this function:



**5.19.3.19 geo2cart()** void geo2cart (  
    double z,  
    double lon,  
    double lat,  
    double \* x )

Convert geolocation to Cartesian coordinates.

Definition at line 3504 of file [jurassic.c](#).

```
03508     {  
03509  
03510     double radius = z + RE;  
03511  
03512     x[0] = radius * cos(lat / 180 * M_PI) * cos(lon / 180 * M_PI);  
03513     x[1] = radius * cos(lat / 180 * M_PI) * sin(lon / 180 * M_PI);  
03514     x[2] = radius * sin(lat / 180 * M_PI);  
03515 }
```

**5.19.3.20 hydrostatic()** void hydrostatic (  
     ctl\_t \* ctl,  
     atm\_t \* atm )

Set hydrostatic equilibrium.

Definition at line 3519 of file jurassic.c.

```

03521     {
03522
03523         const double mmair = 28.96456e-3, mmh2o = 18.0153e-3;
03524
03525         const int ipts = 20;
03526
03527         static int ig_h2o = -999;
03528
03529         double dzmin = 1e99, e = 0;
03530
03531         int ipref = 0;
03532
03533         /* Check reference height... */
03534         if (ctl->hydZ < 0)
03535             return;
03536
03537         /* Determine emitter index of H2O... */
03538         if (ig_h2o == -999)
03539             ig_h2o = find_emitter(ctl, "H2O");
03540
03541         /* Find air parcel next to reference height... */
03542         for (int ip = 0; ip < atm->np; ip++)
03543             if (fabs(atm->z[ip] - ctl->hydZ) < dzmin) {
03544                 dzmin = fabs(atm->z[ip] - ctl->hydZ);
03545                 ipref = ip;
03546             }
03547
03548         /* Upper part of profile... */
03549         for (int ip = ipref + 1; ip < atm->np; ip++) {
03550             double mean = 0;
03551             for (int i = 0; i < ipts; i++) {
03552                 if (ig_h2o >= 0)
03553                     e = LIN(0.0, atm->q[ig_h2o][ip - 1],
03554                             ipts - 1.0, atm->q[ig_h2o][ip], (double) i);
03555                 mean += (e * mmh2o + (1 - e) * mmair)
03556                     * G0 / RI
03557                     / LIN(0.0, atm->t[ip - 1], ipts - 1.0, atm->t[ip], (double) i) / ipts;
03558             }
03559
03560             /* Compute p(z,T)... */
03561             atm->p[ip] =
03562                 exp(log(atm->p[ip - 1]) - mean * 1000 * (atm->z[ip] - atm->z[ip - 1]));
03563         }
03564
03565         /* Lower part of profile... */
03566         for (int ip = ipref - 1; ip >= 0; ip--) {
03567             double mean = 0;
03568             for (int i = 0; i < ipts; i++) {
03569                 if (ig_h2o >= 0)
03570                     e = LIN(0.0, atm->q[ig_h2o][ip + 1],
03571                             ipts - 1.0, atm->q[ig_h2o][ip], (double) i);
03572                 mean += (e * mmh2o + (1 - e) * mmair)
03573                     * G0 / RI
03574                     / LIN(0.0, atm->t[ip + 1], ipts - 1.0, atm->t[ip], (double) i) / ipts;
03575             }
03576
03577             /* Compute p(z,T)... */
03578             atm->p[ip] =
03579                 exp(log(atm->p[ip + 1]) - mean * 1000 * (atm->z[ip] - atm->z[ip + 1]));
03580         }
03581     }

```

Here is the call graph for this function:



**5.19.3.21 idx2name()** void idx2name (  
     ctl\_t \* ctl,  
     int idx,  
     char \* quantity )

Determine name of state vector quantity for given index.

Definition at line 3585 of file [jurassic.c](#).

```

03588     {
03589
03590     if (idx == IDXP)
03591         sprintf(quantity, "PRESSURE");
03592
03593     if (idx == IDXT)
03594         sprintf(quantity, "TEMPERATURE");
03595
03596     for (int ig = 0; ig < ctl->ng; ig++)
03597         if (idx == IDXQ(ig))
03598             sprintf(quantity, "%s", ctl->emitter[ig]);
03599
03600     for (int iw = 0; iw < ctl->nw; iw++)
03601         if (idx == IDXK(iw))
03602             sprintf(quantity, "EXTINCT_WINDOW_%d", iw);
03603
03604     if (idx == IDXCLZ)
03605         sprintf(quantity, "CLOUD_HEIGHT");
03606
03607     if (idx == IDXCLDZ)
03608         sprintf(quantity, "CLOUD_DEPTH");
03609
03610     for (int icl = 0; icl < ctl->ncl; icl++)
03611         if (idx == IDXCLK(icl))
03612             sprintf(quantity, "CLOUD_EXTINCT_%.4f", ctl->clnu[icl]);
03613
03614     if (idx == IDXSFZ)
03615         sprintf(quantity, "SURFACE_HEIGHT");
03616
03617     if (idx == IDXSFP)
03618         sprintf(quantity, "SURFACE_PRESSURE");
03619
03620     if (idx == IDXSFT)
03621         sprintf(quantity, "SURFACE_TEMPERATURE");
03622
03623     for (int isf = 0; isf < ctl->nsf; isf++)
03624         if (idx == IDXSFEP(isf))
03625             sprintf(quantity, "SURFACE_EMISSIVITY_%.4f", ctl->sfnu[isf]);
03626 }
```

**5.19.3.22 init\_srcfunc()** void init\_srcfunc (  
     ctl\_t \* ctl,  
     tbl\_t \* tbl )

Initialize source function table.

Definition at line 3630 of file [jurassic.c](#).

```

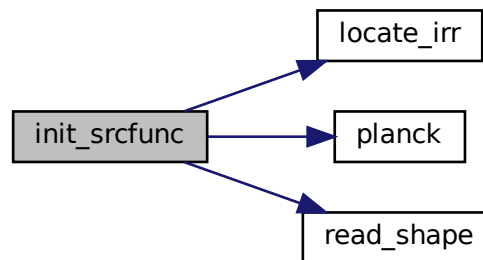
03632     {
03633
03634     char filename[2 * LEN];
03635
03636     double f[NSHAPE], nu[NSHAPE];
03637
03638     int n;
03639
03640     /* Write info... */
03641     LOG(1, "Initialize source function table...");
03642
03643     /* Loop over channels... */
03644     for (int id = 0; id < ctl->nd; id++) {
03645
03646         /* Read filter function... */
```

```

03647     sprintf(filename, "%s_%.4f.filt", ctl->tblbase, ctl->nu[id]);
03648     read_shape(filename, nu, f, &n);
03649
03650     /* Get minimum grid spacing... */
03651     double dnu = 1.0;
03652     for (int i = 1; i < n; i++)
03653         dnu = GSL_MIN(dnu, nu[i] - nu[i - 1]);
03654
03655     /* Compute source function table... */
03656     #pragma omp parallel for default(none) shared(ctl,tbl,id,nu,f,n,dnu)
03657     for (int it = 0; it < TBLNS; it++) {
03658
03659         /* Set temperature... */
03660         tbl->st[it] = LIN(0.0, TMIN, TBLNS - 1.0, TMAX, (double) it);
03661
03662         /* Integrate Planck function... */
03663         double fsum = tbl->sr[it][id] = 0;
03664         for (double fnu = nu[0]; fnu <= nu[n - 1]; fnu += dnu) {
03665             int i = locate_irr(nu, n, fnu);
03666             double ff = LIN(nu[i], f[i], nu[i + 1], f[i + 1], fnu);
03667             fsum += ff;
03668             tbl->sr[it][id] += ff * planck(tbl->st[it], fnu);
03669         }
03670         tbl->sr[it][id] /= fsum;
03671     }
03672 }
03673 }

```

Here is the call graph for this function:



**5.19.3.23 intpol\_atm()** void intpol\_atm (

```

    ctl_t * ctl,
    atm_t * atm,
    double z,
    double * p,
    double * t,
    double * q,
    double * k )

```

Interpolate atmospheric data.

Definition at line 3677 of file [jurassic.c](#).

```

03684     {
03685
03686         /* Get array index... */
03687         int ip = locate_irr(atm->z, atm->np, z);
03688
03689         /* Interpolate... */
03690         *p = EXP(atm->z[ip], atm->p[ip], atm->z[ip + 1], atm->p[ip + 1], z);

```

```

03691  *t = LIN(atm->z[ip], atm->t[ip], atm->z[ip + 1], atm->t[ip + 1], z);
03692  for (int ig = 0; ig < ctl->ng; ig++)
03693      q[ig] =
03694          LIN(atm->z[ip], atm->q[ig][ip], atm->z[ip + 1], atm->q[ig][ip + 1], z);
03695  for (int iw = 0; iw < ctl->nw; iw++)
03696      k[iw] =
03697          LIN(atm->z[ip], atm->k[iw][ip], atm->z[ip + 1], atm->k[iw][ip + 1], z);
03698  }

```

Here is the call graph for this function:



**5.19.3.24 intpol\_tbl()** void intpol\_tbl (

```

    ctl_t * ctl,
    tbl_t * tbl,
    los_t * los,
    int ip,
    double tau_path[ND][NG],
    double tau_seg[ND] )

```

Get transmittance from look-up tables.

Definition at line 3702 of file [jurassic.c](#).

```

03708  {
03709
03710      double eps, u;
03711
03712      /* Loop over channels... */
03713      for (int id = 0; id < ctl->nd; id++) {
03714
03715          /* Initialize... */
03716          tau_seg[id] = 1;
03717
03718          /* Loop over emitters.... */
03719          for (int ig = 0; ig < ctl->ng; ig++) {
03720
03721              /* Check size of table (pressure)... */
03722              if (tbl->np[id][ig] < 30)
03723                  eps = 0;
03724
03725              /* Check transmittance... */
03726              else if (tau_path[id][ig] < 1e-9)
03727                  eps = 1;
03728
03729              /* Interpolate... */
03730              else {
03731
03732                  /* Determine pressure and temperature indices... */
03733                  int ipr = locate_irr(tbl->p[id][ig], tbl->np[id][ig], los->p[ip]);
03734                  int it0 =
03735                      locate_reg(tbl->t[id][ig][ipr], tbl->nt[id][ig][ipr], los->t[ip]);
03736                  int it1 =
03737                      locate_reg(tbl->t[id][ig][ipr + 1], tbl->nt[id][ig][ipr + 1],
03738                          los->t[ip]);
03739
03740                  /* Check size of table (temperature and column density)... */
03741                  if (tbl->nt[id][ig][ipr] < 2 || tbl->nt[id][ig][ipr + 1] < 2
03742                      || tbl->nu[id][ig][ipr][it0] < 2
03743                      || tbl->nu[id][ig][ipr][it0 + 1] < 2
03744                      || tbl->nu[id][ig][ipr + 1][it1] < 2

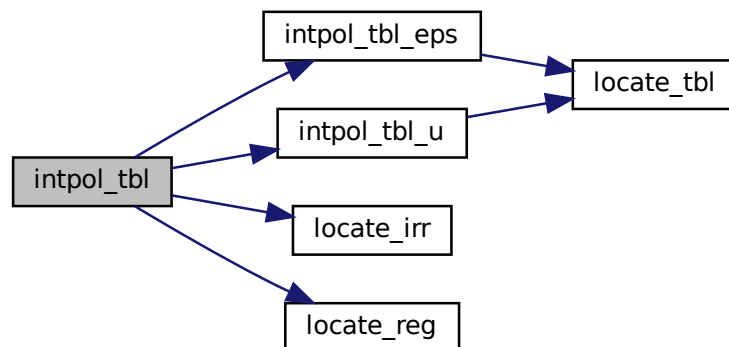
```

```

03745         || tbl->nu[id][ig][ipr + 1][it1 + 1] < 2)
03746         eps = 0;
03747
03748     else {
03749
03750         /* Get emissivities of extended path... */
03751         u = intpol_tbl_u(tbl, ig, id, ipr, it0, 1 - tau_path[id][ig]);
03752         double eps00
03753             = intpol_tbl_eps(tbl, ig, id, ipr, it0, u + los->u[ip][ig]);
03754
03755         u = intpol_tbl_u(tbl, ig, id, ipr, it0 + 1, 1 - tau_path[id][ig]);
03756         double eps01 =
03757             intpol_tbl_eps(tbl, ig, id, ipr, it0 + 1, u + los->u[ip][ig]);
03758
03759         u = intpol_tbl_u(tbl, ig, id, ipr + 1, it1, 1 - tau_path[id][ig]);
03760         double eps10 =
03761             intpol_tbl_eps(tbl, ig, id, ipr + 1, it1, u + los->u[ip][ig]);
03762
03763         u =
03764             intpol_tbl_u(tbl, ig, id, ipr + 1, it1 + 1, 1 - tau_path[id][ig]);
03765         double eps11 =
03766             intpol_tbl_eps(tbl, ig, id, ipr + 1, it1 + 1, u + los->u[ip][ig]);
03767
03768         /* Interpolate with respect to temperature... */
03769         eps00 = LIN(tbl->t[id][ig][ipr][it0], eps00,
03770             tbl->t[id][ig][ipr][it0 + 1], eps01, los->t[ip]);
03771         eps11 = LIN(tbl->t[id][ig][ipr + 1][it1], eps10,
03772             tbl->t[id][ig][ipr + 1][it1 + 1], eps11, los->t[ip]);
03773
03774         /* Interpolate with respect to pressure... */
03775         eps00 = LIN(tbl->p[id][ig][ipr], eps00,
03776             tbl->p[id][ig][ipr + 1], eps11, los->p[ip]);
03777
03778         /* Check emssivity range... */
03779         eps00 = GSL_MAX(GSL_MIN(eps00, 1), 0);
03780
03781         /* Determine segment emissivity... */
03782         eps = 1 - (1 - eps00) / tau_path[id][ig];
03783     }
03784 }
03785
03786 /* Get transmittance of extended path... */
03787 tau_path[id][ig] *= (1 - eps);
03788
03789 /* Get segment transmittance... */
03790 tau_seg[id] *= (1 - eps);
03791 }
03792 }
03793 }

```

Here is the call graph for this function:



**5.19.3.25 intpol\_tbl\_eps()** double intpol\_tbl\_eps (  
tbl\_t \* tbl,  
int ig,  
int id,  
int ip,  
int it,  
double u )

Interpolate emissivity from look-up tables.

Definition at line 3797 of file [jurassic.c](#).

```

03803     {
03804
03805     /* Lower boundary... */
03806     if (u < tbl->u[id][ig][ip][it][0])
03807         return LIN(0, 0, tbl->u[id][ig][ip][it][0], tbl->eps[id][ig][ip][it][0],
03808             u);
03809
03810     /* Upper boundary... */
03811     else if (u > tbl->u[id][ig][ip][it][tbl->nu[id][ig][ip][it] - 1])
03812         return LIN(tbl->u[id][ig][ip][it][tbl->nu[id][ig][ip][it] - 1],
03813             tbl->eps[id][ig][ip][it][tbl->nu[id][ig][ip][it] - 1],
03814             1e30, 1, u);
03815
03816     /* Interpolation... */
03817     else {
03818
03819         /* Get index... */
03820         int idx = locate_tbl(tbl->u[id][ig][ip][it], tbl->nu[id][ig][ip][it], u);
03821
03822         /* Interpolate... */
03823         return LIN(tbl->u[id][ig][ip][it][idx], tbl->eps[id][ig][ip][it][idx],
03824             tbl->u[id][ig][ip][it][idx + 1], tbl->eps[id][ig][ip][it][idx + 1],
03825             u);
03826     }
03827 }
03828 }
```

Here is the call graph for this function:



**5.19.3.26 intpol\_tbl\_u()** double intpol\_tbl\_u (  
tbl\_t \* tbl,  
int ig,  
int id,  
int ip,  
int it,  
double eps )

Interpolate column density from look-up tables.

Definition at line 3832 of file [jurassic.c](#).

```

03838     {
03839
03840     /* Lower boundary... */
```



```

03841  if (eps < tbl->eps[id][ig][ip][it][0])
03842      return LIN(0, 0, tbl->eps[id][ig][ip][it][0], tbl->u[id][ig][ip][it][0],
03843                eps);
03844
03845  /* Upper boundary... */
03846  else if (eps > tbl->eps[id][ig][ip][it][tbl->nu[id][ig][ip][it] - 1])
03847      return LIN(tbl->eps[id][ig][ip][it][tbl->nu[id][ig][ip][it] - 1],
03848                tbl->u[id][ig][ip][it][tbl->nu[id][ig][ip][it] - 1],
03849                1, 1e30, eps);
03850
03851  /* Interpolation... */
03852  else {
03853
03854      /* Get index... */
03855      int idx
03856          = locate_tbl(tbl->eps[id][ig][ip][it], tbl->nu[id][ig][ip][it], eps);
03857
03858      /* Interpolate... */
03859      return
03860          LIN(tbl->eps[id][ig][ip][it][idx], tbl->u[id][ig][ip][it][idx],
03861            tbl->eps[id][ig][ip][it][idx + 1], tbl->u[id][ig][ip][it][idx + 1],
03862            eps);
03863  }
03864 }

```

Here is the call graph for this function:



#### 5.19.3.27 jsec2time() void jsec2time (

```

    double jsec,
    int * year,
    int * mon,
    int * day,
    int * hour,
    int * min,
    int * sec,
    double * remain )

```

Convert seconds to date.

Definition at line 3868 of file [jurassic.c](#).

```

03876  {
03877
03878      struct tm t0, *t1;
03879
03880      t0.tm_year = 100;
03881      t0.tm_mon = 0;
03882      t0.tm_mday = 1;
03883      t0.tm_hour = 0;
03884      t0.tm_min = 0;
03885      t0.tm_sec = 0;
03886
03887      time_t jsec0 = (time_t) jsec + timegm(&t0);
03888      t1 = gmtime(&jsec0);
03889
03890      *year = t1->tm_year + 1900;
03891      *mon = t1->tm_mon + 1;
03892      *day = t1->tm_mday;
03893      *hour = t1->tm_hour;

```

```

03894  *min = t1->tm_min;
03895  *sec = t1->tm_sec;
03896  *remain = jsec - floor(jsec);
03897 }

```

**5.19.3.28 kernel()** void kernel (

```

    ctl_t * ctl,
    atm_t * atm,
    obs_t * obs,
    gsl_matrix * k )

```

Compute Jacobians.

Definition at line 3901 of file [jurassic.c](#).

```

03905  {
03906
03907  atm_t *atml;
03908  obs_t *obs1;
03909
03910  gsl_vector *x0, *x1, *yy0, *yy1;
03911
03912  int *iqa;
03913
03914  /* Get sizes... */
03915  size_t m = k->size1;
03916  size_t n = k->size2;
03917
03918  /* Allocate... */
03919  x0 = gsl_vector_alloc(n);
03920  yy0 = gsl_vector_alloc(m);
03921  ALLOC(iqa, int,
03922        N);
03923
03924  /* Compute radiance for undisturbed atmospheric data... */
03925  formod(ctl, atm, obs);
03926
03927  /* Compose vectors... */
03928  atm2x(ctl, atm, x0, iqa, NULL);
03929  obs2y(ctl, obs, yy0, NULL, NULL);
03930
03931  /* Initialize kernel matrix... */
03932  gsl_matrix_set_zero(k);
03933
03934  /* Loop over state vector elements... */
03935  #pragma omp parallel for default(none) shared(ctl,atm,obs,k,x0,yy0,n,m,iqa) private(x1, yy1, atml,
03936  obs1)
03937  for (size_t j = 0; j < n; j++) {
03938
03939  /* Allocate... */
03940  x1 = gsl_vector_alloc(n);
03941  yy1 = gsl_vector_alloc(m);
03942  ALLOC(atml, atm_t, 1);
03943  ALLOC(obs1, obs_t, 1);
03944
03945  /* Set perturbation size... */
03946  double h;
03947  if (iqa[j] == IDXP)
03948    h = GSL_MAX(fabs(0.01 * gsl_vector_get(x0, j)), 1e-7);
03949  else if (iqa[j] == IDXT)
03950    h = 1.0;
03951  else if (iqa[j] >= IDXQ(0) && iqa[j] < IDXQ(ctl->ng))
03952    h = GSL_MAX(fabs(0.01 * gsl_vector_get(x0, j)), 1e-15);
03953  else if (iqa[j] >= IDXK(0) && iqa[j] < IDXK(ctl->nw))
03954    h = 1e-4;
03955  else if (iqa[j] == IDXCLZ || iqa[j] == IDXCLDZ)
03956    h = 1.0;
03957  else if (iqa[j] >= IDXCLK(0) && iqa[j] < IDXCLK(ctl->ncl))
03958    h = 1e-4;
03959  else if (iqa[j] == IDXSFZ)
03960    h = 0.1;
03961  else if (iqa[j] == IDXSFP)
03962    h = 10.0;
03963  else if (iqa[j] == IDXSFT)
03964    h = 1.0;
03965  else if (iqa[j] >= IDXSFEPS(0) && iqa[j] < IDXSFEPS(ctl->nsf))
03966    h = 1e-2;
03967  else

```

Here is the call graph for this function:



**5.19.3.29 locate\_irr()** int locate\_irr (  
double \* xx,  
int n,  
double x )

Find array index for irregular grid.

Definition at line 4002 of file [jurassic.c](#).

```
04005     {  
04006  
04007     int ilo = 0;  
04008     int ihi = n - 1;  
04009     int i = (ihi + ilo) » 1;  
04010  
04011     if (xx[i] < xx[i + 1])  
04012         while (ihi > ilo + 1) {  
04013         i = (ihi + ilo) » 1;  
04014         if (xx[i] > x)  
04015             ihi = i;  
04016         else  
04017             ilo = i;  
04018     } else  
04019         while (ihi > ilo + 1) {  
04020         i = (ihi + ilo) » 1;  
04021         if (xx[i] <= x)  
04022             ihi = i;  
04023         else  
04024             ilo = i;  
04025     }  
04026  
04027     return ilo;  
04028 }
```

**5.19.3.30 locate\_reg()** int locate\_reg (  
double \* xx,  
int n,  
double x )

Find array index for regular grid.

Definition at line 4032 of file [jurassic.c](#).

```
04035     {  
04036  
04037     /* Calculate index... */  
04038     int i = (int) ((x - xx[0]) / (xx[1] - xx[0]));  
04039  
04040     /* Check range... */  
04041     if (i < 0)  
04042         return 0;  
04043     else if (i > n - 2)  
04044         return n - 2;  
04045     else  
04046         return i;  
04047 }
```

**5.19.3.31 locate\_tbl()** int locate\_tbl (  
float \* xx,  
int n,  
double x )

Find array index in float array.

Definition at line 4051 of file [jurassic.c](#).

```
04054     {  
04055  
04056     int ilo = 0;
```

```

04057     int ihi = n - 1;
04058     int i = (ihi + ilo) » 1;
04059
04060     while (ihi > ilo + 1) {
04061         i = (ihi + ilo) » 1;
04062         if (xx[i] > x)
04063             ihi = i;
04064         else
04065             ilo = i;
04066     }
04067
04068     return ilo;
04069 }

```

**5.19.3.32 obs2y()** size\_t obs2y (  
     ctl\_t \* ctl,  
     obs\_t \* obs,  
     gsl\_vector \* y,  
     int \* ida,  
     int \* ira )

Compose measurement vector.

Definition at line 4073 of file [jurassic.c](#).

```

04078     {
04079
04080     size_t m = 0;
04081
04082     /* Determine measurement vector... */
04083     for (int ir = 0; ir < obs->nr; ir++)
04084         for (int id = 0; id < ctl->nd; id++)
04085             if (gsl_finite(obs->rad[id][ir])) {
04086                 if (y != NULL)
04087                     gsl_vector_set(y, m, obs->rad[id][ir]);
04088                 if (ida != NULL)
04089                     ida[m] = id;
04090                 if (ira != NULL)
04091                     ira[m] = ir;
04092                 m++;
04093             }
04094
04095     return m;
04096 }

```

**5.19.3.33 planck()** double planck (  
     double t,  
     double nu )

Compute Planck function.

Definition at line 4100 of file [jurassic.c](#).

```

04102     {
04103
04104     return C1 * POW3(nu) / gsl_expm1(C2 * nu / t);
04105 }

```

**5.19.3.34 raytrace()** void raytrace (

```

    ctl_t * ctl,
    atm_t * atm,
    obs_t * obs,
    los_t * los,
    int ir )

```

Do ray-tracing to determine LOS.

Definition at line 4109 of file [jurassic.c](#).

```

04114     {
04115
04116     const double h = 0.02, zrefrac = 60;
04117
04118     double ds, ex0[3], ex1[3], k[NW], lat, lon, n, ng[3], norm,
04119         p, q[NG], t, x[3], xh[3], xobs[3], xvp[3], z = 1e99, zmax, zmin;
04120
04121     int stop = 0;
04122
04123     /* Initialize... */
04124     los->np = 0;
04125     los->sft = -999;
04126     obs->tpz[ir] = obs->vpz[ir];
04127     obs->tplon[ir] = obs->vplon[ir];
04128     obs->tplat[ir] = obs->vplat[ir];
04129
04130     /* Get altitude range of atmospheric data... */
04131     gsl_stats_minmax(&zmin, &zmax, atm->z, 1, (size_t) atm->np);
04132     if (ctl->nsf > 0) {
04133         zmin = GSL_MAX(atm->sfz, zmin);
04134         if (atm->sfp > 0) {
04135             int ip = locate_irr(atm->p, atm->np, atm->sfp);
04136             double zip = LIN(log(atm->p[ip]), atm->z[ip],
04137                 log(atm->p[ip + 1]), atm->z[ip + 1], log(atm->sfp));
04138             zmin = GSL_MAX(zip, zmin);
04139         }
04140     }
04141
04142     /* Check observer altitude... */
04143     if (obs->obsz[ir] < zmin)
04144         ERRMSG("Observer below surface!");
04145
04146     /* Check view point altitude... */
04147     if (obs->vpz[ir] > zmax)
04148         return;
04149
04150     /* Determine Cartesian coordinates for observer and view point... */
04151     geo2cart(obs->obsz[ir], obs->obslon[ir], obs->obslat[ir], xobs);
04152     geo2cart(obs->vpz[ir], obs->vplon[ir], obs->vplat[ir], xvp);
04153
04154     /* Determine initial tangent vector... */
04155     for (int i = 0; i < 3; i++)
04156         ex0[i] = xvp[i] - xobs[i];
04157     norm = NORM(ex0);
04158     for (int i = 0; i < 3; i++)
04159         ex0[i] /= norm;
04160
04161     /* Observer within atmosphere... */
04162     for (int i = 0; i < 3; i++)
04163         x[i] = xobs[i];
04164
04165     /* Observer above atmosphere (search entry point)... */
04166     if (obs->obsz[ir] > zmax) {
04167         double dmax = norm, dmin = 0;
04168         while (fabs(dmin - dmax) > 0.001) {
04169             double d = (dmax + dmin) / 2;
04170             for (int i = 0; i < 3; i++)
04171                 x[i] = xobs[i] + d * ex0[i];
04172             cart2geo(x, &z, &lon, &lat);
04173             if (z <= zmax && z > zmax - 0.001)
04174                 break;
04175             if (z < zmax - 0.0005)
04176                 dmax = d;
04177             else
04178                 dmin = d;
04179         }
04180     }
04181
04182     /* Ray-tracing... */
04183     while (1) {
04184
04185         /* Set step length... */
04186         ds = ctl->rayds;

```

```

04187     if (ctl->raydz > 0) {
04188         norm = NORM(x);
04189         for (int i = 0; i < 3; i++)
04190             xh[i] = x[i] / norm;
04191         double cosa = fabs(DOTP(ex0, xh));
04192         if (cosa != 0)
04193             ds = GSL_MIN(ctl->rayds, ctl->raydz / cosa);
04194     }
04195
04196     /* Determine geolocation... */
04197     cart2geo(x, &z, &lon, &lat);
04198
04199     /* Check if LOS hits the ground or has left atmosphere... */
04200     if (z < zmin || z > zmax) {
04201         stop = (z < zmin ? 2 : 1);
04202         double frac =
04203             ((z <
04204              zmin ? zmin : zmax) - los->z[los->np - 1]) / (z - los->z[los->np -
04205                                                         1]);
04206         geo2cart(los->z[los->np - 1], los->lon[los->np - 1],
04207                 los->lat[los->np - 1], xh);
04208         for (int i = 0; i < 3; i++)
04209             x[i] = xh[i] + frac * (x[i] - xh[i]);
04210         cart2geo(x, &z, &lon, &lat);
04211         los->ds[los->np - 1] = ds * frac;
04212         ds = 0;
04213     }
04214
04215     /* Interpolate atmospheric data... */
04216     intpol_atm(ctl, atm, z, &p, &t, q, k);
04217
04218     /* Save data... */
04219     los->lon[los->np] = lon;
04220     los->lat[los->np] = lat;
04221     los->z[los->np] = z;
04222     los->p[los->np] = p;
04223     los->t[los->np] = t;
04224     for (int ig = 0; ig < ctl->ng; ig++)
04225         los->q[los->np][ig] = q[ig];
04226     for (int id = 0; id < ctl->nd; id++)
04227         los->k[los->np][id] = k[ctl->>window[id]];
04228     los->ds[los->np] = ds;
04229
04230     /* Add cloud extinction... */
04231     if (ctl->ncl > 0 && atm->cldz > 0) {
04232         double aux = exp(-0.5 * POW2((z - atm->clz) / atm->cldz));
04233         for (int id = 0; id < ctl->nd; id++) {
04234             int icl = locate_irr(ctl->clnu, ctl->ncl, ctl->nu[id]);
04235             los->k[los->np][id]
04236                 += aux * LIN(ctl->clnu[icl], atm->clk[icl],
04237                             ctl->clnu[icl + 1], atm->clk[icl + 1], ctl->nu[id]);
04238         }
04239     }
04240
04241     /* Increment and check number of LOS points... */
04242     if ((++los->np) > NLOS)
04243         ERRMSG("Too many LOS points!");
04244
04245     /* Check stop flag... */
04246     if (stop) {
04247
04248         /* Set surface temperature... */
04249         if (ctl->nsf > 0 && atm->sft > 0)
04250             t = atm->sft;
04251         los->sft = (stop == 2 ? t : -999);
04252
04253         /* Set surface emissivity... */
04254         for (int id = 0; id < ctl->nd; id++) {
04255             los->sfeps[id] = 1.0;
04256             if (ctl->nsf > 0) {
04257                 int isf = locate_irr(ctl->sfnu, ctl->nsf, ctl->nu[id]);
04258                 los->sfeps[id] = LIN(ctl->sfnu[isf], atm->sfeps[isf],
04259                                     ctl->sfnu[isf + 1], atm->sfeps[isf + 1],
04260                                     ctl->nu[id]);
04261             }
04262         }
04263
04264         /* Leave raytracer... */
04265         break;
04266     }
04267
04268     /* Determine refractivity... */
04269     if (ctl->refrac && z <= zrefrac)
04270         n = 1 + refractivity(p, t);
04271     else
04272         n = 1;
04273

```

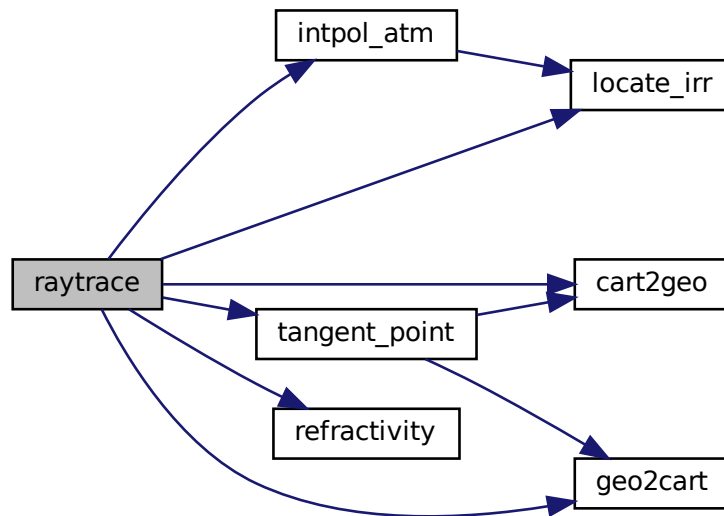
```

04274      /* Construct new tangent vector (first term)... */
04275      for (int i = 0; i < 3; i++)
04276          exl[i] = ex0[i] * n;
04277
04278      /* Compute gradient of refractivity... */
04279      if (ctl->refrac && z <= zrefrac) {
04280          for (int i = 0; i < 3; i++)
04281              xh[i] = x[i] + 0.5 * ds * ex0[i];
04282          cart2geo(xh, &z, &lon, &lat);
04283          intpol_atm(ctl, atm, z, &p, &t, q, k);
04284          n = refractivity(p, t);
04285          for (int i = 0; i < 3; i++) {
04286              xh[i] += h;
04287              cart2geo(xh, &z, &lon, &lat);
04288              intpol_atm(ctl, atm, z, &p, &t, q, k);
04289              ng[i] = (refractivity(p, t) - n) / h;
04290              xh[i] -= h;
04291          }
04292      } else
04293          for (int i = 0; i < 3; i++)
04294              ng[i] = 0;
04295
04296      /* Construct new tangent vector (second term)... */
04297      for (int i = 0; i < 3; i++)
04298          exl[i] += ds * ng[i];
04299
04300      /* Normalize new tangent vector... */
04301      norm = NORM(exl);
04302      for (int i = 0; i < 3; i++)
04303          exl[i] /= norm;
04304
04305      /* Determine next point of LOS... */
04306      for (int i = 0; i < 3; i++)
04307          x[i] += 0.5 * ds * (ex0[i] + exl[i]);
04308
04309      /* Copy tangent vector... */
04310      for (int i = 0; i < 3; i++)
04311          ex0[i] = exl[i];
04312  }
04313
04314      /* Get tangent point (to be done before changing segment lengths!)... */
04315      tangent_point(los, &obs->tpz[ir], &obs->tplon[ir], &obs->tplat[ir]);
04316
04317      /* Change segment lengths according to trapezoid rule... */
04318      for (int ip = los->np - 1; ip >= 1; ip--)
04319          los->ds[ip] = 0.5 * (los->ds[ip - 1] + los->ds[ip]);
04320      los->ds[0] *= 0.5;
04321
04322      /* Compute column density... */
04323      for (int ip = 0; ip < los->np; ip++)
04324          for (int ig = 0; ig < ctl->ng; ig++)
04325              los->u[ip][ig] = 10 * los->q[ip][ig] * los->p[ip]
04326                  / (KB * los->t[ip]) * los->ds[ip];
04327  }

```



Here is the call graph for this function:



**5.19.3.35 read\_atm()** void read\_atm (  
 const char \* dirname,  
 const char \* filename,  
 ctl\_t \* ctl,  
 atm\_t \* atm )

Read atmospheric data.

Definition at line 4331 of file [jurassic.c](#).

```

04335     {
04336
04337     FILE *in;
04338
04339     char file[LEN], line[LEN], *tok;
04340
04341     /* Init... */
04342     atm->np = 0;
04343
04344     /* Set filename... */
04345     if (dirname != NULL)
04346         sprintf(file, "%s/%s", dirname, filename);
04347     else
04348         sprintf(file, "%s", filename);
04349
04350     /* Write info... */
04351     LOG(1, "Read atmospheric data: %s", file);
04352
04353     /* Open file... */
04354     if (!(in = fopen(file, "r")))
04355         ERRMSG("Cannot open file!");
04356
04357     /* Read line... */
04358     while (fgets(line, LEN, in)) {
04359
04360         /* Read data... */
04361         TOK(line, tok, "%lg", atm->time[atm->np]);
04362         TOK(NULL, tok, "%lg", atm->z[atm->np]);
  
```

```

04363     TOK(NULL, tok, "%lg", atm->lon[atm->np]);
04364     TOK(NULL, tok, "%lg", atm->lat[atm->np]);
04365     TOK(NULL, tok, "%lg", atm->p[atm->np]);
04366     TOK(NULL, tok, "%lg", atm->t[atm->np]);
04367     for (int ig = 0; ig < ctl->ng; ig++)
04368         TOK(NULL, tok, "%lg", atm->q[ig][atm->np]);
04369     for (int iw = 0; iw < ctl->nw; iw++)
04370         TOK(NULL, tok, "%lg", atm->k[iw][atm->np]);
04371     if (ctl->ncl > 0 && atm->np == 0) {
04372         TOK(NULL, tok, "%lg", atm->clz);
04373         TOK(NULL, tok, "%lg", atm->cldz);
04374         for (int icl = 0; icl < ctl->ncl; icl++)
04375             TOK(NULL, tok, "%lg", atm->clk[icl]);
04376     }
04377     if (ctl->nsf > 0 && atm->np == 0) {
04378         TOK(NULL, tok, "%lg", atm->sfz);
04379         TOK(NULL, tok, "%lg", atm->sfp);
04380         TOK(NULL, tok, "%lg", atm->sft);
04381         for (int isf = 0; isf < ctl->nsf; isf++)
04382             TOK(NULL, tok, "%lg", atm->sfeps[isf]);
04383     }
04384
04385     /* Increment data point counter... */
04386     if ((++atm->np) > NP)
04387         ERRMSG("Too many data points!");
04388 }
04389
04390 /* Close file... */
04391 fclose(in);
04392
04393 /* Check number of points... */
04394 if (atm->np < 1)
04395     ERRMSG("Could not read any data!");
04396 }

```

**5.19.3.36 read\_ctl()** void read\_ctl (  
     int argc,  
     char \* argv[],  
     ctl\_t \* ctl )

Read forward model control parameters.

Definition at line 4400 of file [jurassic.c](#).

```

04403     {
04404
04405     /* Write info... */
04406     LOG(1, "\nJuelich Rapid Spectral Simulation Code (JURASSIC)\n"
04407          "(executable: %s | version: %s | compiled: %s, %s)\n",
04408          argv[0], VERSION, __DATE__, __TIME__);
04409
04410     /* Emitters... */
04411     ctl->ng = (int) scan_ctl(argc, argv, "NG", -1, "0", NULL);
04412     if (ctl->ng < 0 || ctl->ng > NG)
04413         ERRMSG("Set 0 <= NG <= MAX!");
04414     for (int ig = 0; ig < ctl->ng; ig++)
04415         scan_ctl(argc, argv, "EMITTER", ig, "", ctl->emitter[ig]);
04416
04417     /* Radiance channels... */
04418     ctl->nd = (int) scan_ctl(argc, argv, "ND", -1, "0", NULL);
04419     if (ctl->nd < 0 || ctl->nd > ND)
04420         ERRMSG("Set 0 <= ND <= MAX!");
04421     for (int id = 0; id < ctl->nd; id++)
04422         ctl->nu[id] = scan_ctl(argc, argv, "NU", id, "", NULL);
04423
04424     /* Spectral windows... */
04425     ctl->nw = (int) scan_ctl(argc, argv, "NW", -1, "1", NULL);
04426     if (ctl->nw < 0 || ctl->nw > NW)
04427         ERRMSG("Set 0 <= NW <= MAX!");
04428     for (int id = 0; id < ctl->nd; id++)
04429         ctl->>window[id] = (int) scan_ctl(argc, argv, "WINDOW", id, "0", NULL);
04430
04431     /* Cloud data... */
04432     ctl->ncl = (int) scan_ctl(argc, argv, "NCL", -1, "0", NULL);
04433     if (ctl->ncl < 0 || ctl->ncl > NCL)
04434         ERRMSG("Set 0 <= NCL <= MAX!");
04435     if (ctl->ncl == 1)
04436         ERRMSG("Set NCL > 1!");
04437     for (int icl = 0; icl < ctl->ncl; icl++)

```

```

04438     ctl->clnu[icl] = scan_ctl(argc, argv, "CLNU", icl, "", NULL);
04439
04440     /* Surface data... */
04441     ctl->nsf = (int) scan_ctl(argc, argv, "NSF", -1, "0", NULL);
04442     if (ctl->nsf < 0 || ctl->nsf > NSF)
04443         ERRMSG("Set 0 <= NSF <= MAX!");
04444     if (ctl->nsf == 1)
04445         ERRMSG("Set NSF > 1!");
04446     for (int isf = 0; isf < ctl->nsf; isf++)
04447         ctl->sfnu[isf] = scan_ctl(argc, argv, "SFNU", isf, "", NULL);
04448     ctl->sftype = (int) scan_ctl(argc, argv, "SFTYPE", -1, "2", NULL);
04449     if (ctl->sftype < 0 || ctl->sftype > 3)
04450         ERRMSG("Set 0 <= SFTYPE <= 3!");
04451     ctl->sfsza = scan_ctl(argc, argv, "SFSZA", -1, "-999", NULL);
04452
04453     /* Emissivity look-up tables... */
04454     scan_ctl(argc, argv, "TBLBASE", -1, "-", ctl->tblbase);
04455     ctl->tblfmt = (int) scan_ctl(argc, argv, "TBLFMT", -1, "1", NULL);
04456
04457     /* Hydrostatic equilibrium... */
04458     ctl->hydZ = scan_ctl(argc, argv, "HYDZ", -1, "-999", NULL);
04459
04460     /* Continua... */
04461     ctl->ctm_co2 = (int) scan_ctl(argc, argv, "CTM_CO2", -1, "1", NULL);
04462     ctl->ctm_h2o = (int) scan_ctl(argc, argv, "CTM_H2O", -1, "1", NULL);
04463     ctl->ctm_n2 = (int) scan_ctl(argc, argv, "CTM_N2", -1, "1", NULL);
04464     ctl->ctm_o2 = (int) scan_ctl(argc, argv, "CTM_O2", -1, "1", NULL);
04465
04466     /* Ray-tracing... */
04467     ctl->refrac = (int) scan_ctl(argc, argv, "REFRAC", -1, "1", NULL);
04468     ctl->rayds = scan_ctl(argc, argv, "RAYDS", -1, "10", NULL);
04469     ctl->raydz = scan_ctl(argc, argv, "RAYDZ", -1, "0.1", NULL);
04470
04471     /* Field of view... */
04472     scan_ctl(argc, argv, "FOV", -1, "-", ctl->fov);
04473
04474     /* Retrieval interface... */
04475     ctl->retp_zmin = scan_ctl(argc, argv, "RETP_ZMIN", -1, "-999", NULL);
04476     ctl->retp_zmax = scan_ctl(argc, argv, "RETP_ZMAX", -1, "-999", NULL);
04477     ctl->rett_zmin = scan_ctl(argc, argv, "RETT_ZMIN", -1, "-999", NULL);
04478     ctl->rett_zmax = scan_ctl(argc, argv, "RETT_ZMAX", -1, "-999", NULL);
04479     for (int ig = 0; ig < ctl->ng; ig++) {
04480         ctl->retq_zmin[ig] = scan_ctl(argc, argv, "RETQ_ZMIN", ig, "-999", NULL);
04481         ctl->retq_zmax[ig] = scan_ctl(argc, argv, "RETQ_ZMAX", ig, "-999", NULL);
04482     }
04483     for (int iw = 0; iw < ctl->nw; iw++) {
04484         ctl->retk_zmin[iw] = scan_ctl(argc, argv, "RETK_ZMIN", iw, "-999", NULL);
04485         ctl->retk_zmax[iw] = scan_ctl(argc, argv, "RETK_ZMAX", iw, "-999", NULL);
04486     }
04487     ctl->ret_clz = (int) scan_ctl(argc, argv, "RET_CLZ", -1, "0", NULL);
04488     ctl->ret_cldz = (int) scan_ctl(argc, argv, "RET_CLDZ", -1, "0", NULL);
04489     ctl->ret_clk = (int) scan_ctl(argc, argv, "RET_CLK", -1, "0", NULL);
04490     ctl->ret_sfz = (int) scan_ctl(argc, argv, "RET_SFZ", -1, "0", NULL);
04491     ctl->ret_sfp = (int) scan_ctl(argc, argv, "RET_SFP", -1, "0", NULL);
04492     ctl->ret_sft = (int) scan_ctl(argc, argv, "RET_SFT", -1, "0", NULL);
04493     ctl->ret_sfeps = (int) scan_ctl(argc, argv, "RET_SFEPS", -1, "0", NULL);
04494
04495     /* Output flags... */
04496     ctl->write_bbt = (int) scan_ctl(argc, argv, "WRITE_BBT", -1, "0", NULL);
04497     ctl->write_matrix =
04498         (int) scan_ctl(argc, argv, "WRITE_MATRIX", -1, "0", NULL);
04499
04500     /* External forward models... */
04501     ctl->formod = (int) scan_ctl(argc, argv, "FORMOD", -1, "1", NULL);
04502     scan_ctl(argc, argv, "RFMBIN", -1, "-", ctl->rfrm-bin);
04503     scan_ctl(argc, argv, "RFMHIT", -1, "-", ctl->rfrm-hit);
04504     for (int ig = 0; ig < ctl->ng; ig++)
04505         scan_ctl(argc, argv, "RFMXSC", ig, "-", ctl->rfrm-xsc[ig]);
04506 }

```

Here is the call graph for this function:



**5.19.3.37 read\_matrix()** void read\_matrix (  
     const char \* *dirname*,  
     const char \* *filename*,  
     gsl\_matrix \* *matrix* )

Read matrix.

Definition at line 4510 of file [jurassic.c](#).

```
04513     {
04514
04515     FILE *in;
04516
04517     char dum[LEN], file[LEN], line[LEN];
04518
04519     double value;
04520
04521     int i, j;
04522
04523     /* Set filename... */
04524     if (dirname != NULL)
04525         sprintf(file, "%s/%s", dirname, filename);
04526     else
04527         sprintf(file, "%s", filename);
04528
04529     /* Write info... */
04530     LOG(1, "Read matrix: %s", file);
04531
04532     /* Open file... */
04533     if (!(in = fopen(file, "r")))
04534         ERRMSG("Cannot open file!");
04535
04536     /* Read data... */
04537     gsl_matrix_set_zero(matrix);
04538     while (fgets(line, LEN, in))
04539         if (sscanf(line, "%d %s %s %s %s %d %s %s %s %s %s %lg",
04540                 &i, dum, dum, dum, dum, dum,
04541                 &j, dum, dum, dum, dum, dum, &value) == 13)
04542             gsl_matrix_set(matrix, (size_t) i, (size_t) j, value);
04543
04544     /* Close file... */
04545     fclose(in);
04546 }
```

**5.19.3.38 read\_obs()** void read\_obs (  
     const char \* *dirname*,  
     const char \* *filename*,  
     ctl\_t \* *ctl*,  
     obs\_t \* *obs* )

Read observation data.

Definition at line 4550 of file [jurassic.c](#).

```
04554     {
04555
04556     FILE *in;
04557
04558     char file[LEN], line[LEN], *tok;
04559
04560     /* Init... */
04561     obs->nr = 0;
04562
04563     /* Set filename... */
04564     if (dirname != NULL)
04565         sprintf(file, "%s/%s", dirname, filename);
04566     else
04567         sprintf(file, "%s", filename);
04568
04569     /* Write info... */
04570     LOG(1, "Read observation data: %s", file);
```

```

04571
04572  /* Open file... */
04573  if (!(in = fopen(file, "r")))
04574      ERRMSG("Cannot open file!");
04575
04576  /* Read line... */
04577  while (fgets(line, LEN, in)) {
04578
04579      /* Read data... */
04580      TOK(line, tok, "%lg", obs->time[obs->nr]);
04581      TOK(NULL, tok, "%lg", obs->obsz[obs->nr]);
04582      TOK(NULL, tok, "%lg", obs->obslon[obs->nr]);
04583      TOK(NULL, tok, "%lg", obs->obslat[obs->nr]);
04584      TOK(NULL, tok, "%lg", obs->vpz[obs->nr]);
04585      TOK(NULL, tok, "%lg", obs->vplon[obs->nr]);
04586      TOK(NULL, tok, "%lg", obs->vplat[obs->nr]);
04587      TOK(NULL, tok, "%lg", obs->tpz[obs->nr]);
04588      TOK(NULL, tok, "%lg", obs->tplon[obs->nr]);
04589      TOK(NULL, tok, "%lg", obs->tplat[obs->nr]);
04590      for (int id = 0; id < ctl->nd; id++)
04591          TOK(NULL, tok, "%lg", obs->rad[id][obs->nr]);
04592      for (int id = 0; id < ctl->nd; id++)
04593          TOK(NULL, tok, "%lg", obs->tau[id][obs->nr]);
04594
04595      /* Increment counter... */
04596      if ((++obs->nr) > NR)
04597          ERRMSG("Too many rays!");
04598  }
04599
04600  /* Close file... */
04601  fclose(in);
04602
04603  /* Check number of points... */
04604  if (obs->nr < 1)
04605      ERRMSG("Could not read any data!");
04606 }

```

**5.19.3.39 read\_obs\_rfm()** double read\_obs\_rfm (

```

    const char * basename,
    double z,
    double * nu,
    double * f,
    int n )

```

Read observation data in RFM format.

Definition at line 4610 of file [jurassic.c](#).

```

04615  {
04616
04617  FILE *in;
04618
04619  char filename[LEN];
04620
04621  double filt, fsum = 0, nu2[NSHAPE], *nurfm, *rad, radsum = 0;
04622
04623  int i, idx, ipt, npts;
04624
04625  /* Allocate... */
04626  ALLOC(nurfm, double,
04627        RFMNPTS);
04628  ALLOC(rad, double,
04629        RFMNPTS);
04630
04631  /* Search RFM spectrum... */
04632  sprintf(filename, "%s_%05d.asc", basename, (int) (z * 1000));
04633  if (!(in = fopen(filename, "r"))) {
04634      sprintf(filename, "%s_%05d.asc", basename, (int) (z * 1000) + 1);
04635      if (!(in = fopen(filename, "r")))
04636          ERRMSG("Cannot find RFM data file!");
04637  }
04638  fclose(in);
04639
04640  /* Read RFM spectrum... */
04641  read_rfm_spec(filename, nurfm, rad, &npts);
04642
04643  /* Set wavenumbers... */
04644  nu2[0] = nu[0];

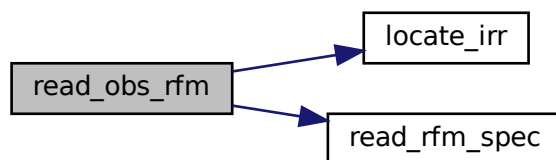
```

```

04645     nu2[n - 1] = nu[n - 1];
04646     for (i = 1; i < n - 1; i++)
04647         nu2[i] = LIN(0.0, nu2[0], n - 1.0, nu2[n - 1], i);
04648
04649     /* Convolute... */
04650     for (ipts = 0; ipts < npts; ipts++)
04651         if (nurfm[ipts] >= nu2[0] && nurfm[ipts] <= nu2[n - 1]) {
04652             idx = locate_irr(nu2, n, nurfm[ipts]);
04653             filt = LIN(nu2[idx], f[idx], nu2[idx + 1], f[idx + 1], nurfm[ipts]);
04654             fsum += filt;
04655             radsum += filt * rad[ipts];
04656         }
04657
04658     /* Free... */
04659     free(nurfm);
04660     free(rad);
04661
04662     /* Return radiance... */
04663     return radsum / fsum;
04664 }

```

Here is the call graph for this function:



**5.19.3.40 read\_rfm\_spec()** void read\_rfm\_spec (

```

    const char * filename,
    double * nu,
    double * rad,
    int * npts )

```

Read RFM spectrum.

Definition at line 4668 of file [jurassic.c](#).

```

04672     {
04673
04674         FILE *in;
04675
04676         char line[RFMLINE], *tok;
04677
04678         double dnu, nu0, nul;
04679
04680         int i, ipts = 0;
04681
04682         /* Write info... */
04683         printf("Read RFM data: %s\n", filename);
04684
04685         /* Open file... */
04686         if (!(in = fopen(filename, "r")))
04687             ERRMSG("Cannot open file!");
04688
04689         /* Read header..... */
04690         for (i = 0; i < 4; i++)
04691             if (fgets(line, RFMLINE, in) == NULL)
04692                 ERRMSG("Error while reading file header!");
04693         sscanf(line, "%d %lg %lg %lg", npts, &nu0, &dnu, &nul);
04694         if (*npts > RFMNPTS)

```

```

04695     ERRMSG("Too many spectral grid points!");
04696
04697     /* Read radiance data... */
04698     while (fgets(line, RFMLINE, in) && ipts < *npts - 1) {
04699         if ((tok = strtok(line, " \t\n")) != NULL)
04700             if (sscanf(tok, "%lg", &rad[ipts]) == 1)
04701                 ipts++;
04702         while ((tok = strtok(NULL, " \t\n")) != NULL)
04703             if (sscanf(tok, "%lg", &rad[ipts]) == 1)
04704                 ipts++;
04705     }
04706     if (ipts != *npts)
04707         ERRMSG("Error while reading RFM data!");
04708
04709     /* Compute wavenumbers... */
04710     for (ipts = 0; ipts < *npts; ipts++)
04711         nu[ipts] = LIN(0.0, nu0, (double) (*npts - 1), nul, (double) ipts);
04712
04713     /* Close file... */
04714     fclose(in);
04715 }

```

**5.19.3.41 read\_shape()** void read\_shape (  
     const char \* filename,  
     double \* x,  
     double \* y,  
     int \* n )

Read shape function.

Definition at line 4719 of file [jurassic.c](#).

```

04723     {
04724
04725     FILE *in;
04726
04727     char line[LEN];
04728
04729     /* Write info... */
04730     LOG(1, "Read shape function: %s", filename);
04731
04732     /* Open file... */
04733     if (!(in = fopen(filename, "r")))
04734         ERRMSG("Cannot open file!");
04735
04736     /* Read data... */
04737     *n = 0;
04738     while (fgets(line, LEN, in))
04739         if (sscanf(line, "%lg %lg", &x[*n], &y[*n]) == 2)
04740             if (++(*n) > NSHAPE)
04741                 ERRMSG("Too many data points!");
04742
04743     /* Check number of points... */
04744     if (*n < 1)
04745         ERRMSG("Could not read any data!");
04746
04747     /* Close file... */
04748     fclose(in);
04749 }

```

**5.19.3.42 read\_tbl()** void read\_tbl (  
     ctl\_t \* ctl,  
     tbl\_t \* tbl )

Read look-up table data.

Definition at line 4753 of file [jurassic.c](#).

```

04755     {
04756
04757     FILE *in;

```

```

04758
04759 char filename[2 * LEN], line[LEN];
04760
04761 double eps, press, temp, u;
04762
04763 /* Loop over trace gases and channels... */
04764 for (int id = 0; id < ctl->nd; id++)
04765     for (int ig = 0; ig < ctl->ng; ig++) {
04766
04767         /* Initialize... */
04768         tbl->np[id][ig] = -1;
04769         double eps_old = -999;
04770         double press_old = -999;
04771         double temp_old = -999;
04772         double u_old = -999;
04773
04774         /* Set filename... */
04775         sprintf(filename, "%s_%.4f_%.s.%s", ctl->tblbase,
04776             ctl->nu[id], ctl->emitter[ig],
04777             ctl->tblfmt == 1 ? "tab" : "bin");
04778
04779         /* Write info... */
04780         LOG(1, "Read emissivity table: %s", filename);
04781
04782         /* Try to open file... */
04783         if (!(in = fopen(filename, "r"))) {
04784             WARN("Missing emissivity table: %s", filename);
04785             continue;
04786         }
04787
04788         /* Read ASCII tables... */
04789         if (ctl->tblfmt == 1) {
04790
04791             /* Read data... */
04792             while (fgets(line, LEN, in)) {
04793
04794                 /* Parse line... */
04795                 if (sscanf(line, "%lg %lg %lg %lg", &press, &temp, &u, &eps) != 4)
04796                     continue;
04797
04798                 /* Check ranges... */
04799                 if (u < 0 || u > 1e30 || eps < 0 || eps > 1)
04800                     continue;
04801
04802                 /* Determine pressure index... */
04803                 if (press != press_old) {
04804                     press_old = press;
04805                     if (++tbl->np[id][ig] >= TBLNP)
04806                         ERRMSG("Too many pressure levels!");
04807                     tbl->nt[id][ig][tbl->np[id][ig]] = -1;
04808                 }
04809
04810                 /* Determine temperature index... */
04811                 if (temp != temp_old) {
04812                     temp_old = temp;
04813                     if (++tbl->nt[id][ig][tbl->np[id][ig]] >= TBLNT)
04814                         ERRMSG("Too many temperatures!");
04815                     tbl->nu[id][ig][tbl->np[id][ig]]
04816                         [tbl->nt[id][ig][tbl->np[id][ig]]] = -1;
04817                 }
04818
04819                 /* Determine column density index... */
04820                 if ((eps > eps_old && u > u_old) || tbl->nu[id][ig][tbl->np[id][ig]]
04821                     [tbl->nt[id][ig][tbl->np[id][ig]]] < 0) {
04822                     eps_old = eps;
04823                     u_old = u;
04824                     if (++tbl->nu[id][ig][tbl->np[id][ig]]
04825                         [tbl->nt[id][ig][tbl->np[id][ig]]] >= TBLNU) {
04826                         tbl->nu[id][ig][tbl->np[id][ig]]
04827                             [tbl->nt[id][ig][tbl->np[id][ig]]]--;
04828                         continue;
04829                     }
04830                 }
04831
04832                 /* Store data... */
04833                 tbl->p[id][ig][tbl->np[id][ig]] = press;
04834                 tbl->t[id][ig][tbl->np[id][ig]][tbl->nt[id][ig][tbl->np[id][ig]]]
04835                     = temp;
04836                 tbl->u[id][ig][tbl->np[id][ig]][tbl->nt[id][ig][tbl->np[id][ig]]]
04837                     [tbl->nu[id][ig][tbl->np[id][ig]]]
04838                     [tbl->nt[id][ig][tbl->np[id][ig]]] = (float) u;
04839                 tbl->eps[id][ig][tbl->np[id][ig]][tbl->nt[id][ig][tbl->np[id][ig]]]
04840                     [tbl->nu[id][ig][tbl->np[id][ig]]]
04841                     [tbl->nt[id][ig][tbl->np[id][ig]]] = (float) eps;
04842             }
04843
04844             /* Increment counters... */

```



```

04845     tbl->np[id][ig]++;
04846     for (int ip = 0; ip < tbl->np[id][ig]; ip++) {
04847         tbl->nt[id][ig][ip]++;
04848         for (int it = 0; it < tbl->nt[id][ig][ip]; it++)
04849             tbl->nu[id][ig][ip][it]++;
04850     }
04851 }
04852
04853 /* Read binary data... */
04854 else if (ctl->tblfmt == 2) {
04855
04856     /* Read data... */
04857     FREAD(&tbl->np[id][ig], int,
04858          1,
04859          in);
04860     if (tbl->np[id][ig] > TBLNP)
04861         ERRMSG("Too many pressure levels!");
04862     FREAD(tbl->p[id][ig], double,
04863          (size_t) tbl->np[id][ig],
04864          in);
04865     for (int ip = 0; ip < tbl->np[id][ig]; ip++) {
04866         FREAD(&tbl->nt[id][ig][ip], int,
04867              1,
04868              in);
04869         if (tbl->nt[id][ig][ip] > TBLNT)
04870             ERRMSG("Too many temperatures!");
04871         FREAD(tbl->t[id][ig][ip], double,
04872              (size_t) tbl->nt[id][ig][ip],
04873              in);
04874         for (int it = 0; it < tbl->nt[id][ig][ip]; it++) {
04875             FREAD(&tbl->nu[id][ig][ip][it], int,
04876                  1,
04877                  in);
04878             if (tbl->nu[id][ig][ip][it] > TBLNU)
04879                 ERRMSG("Too many column densities!");
04880             FREAD(tbl->u[id][ig][ip][it], float,
04881                  (size_t) tbl->nu[id][ig][ip][it],
04882                  in);
04883             FREAD(tbl->eps[id][ig][ip][it], float,
04884                  (size_t) tbl->nu[id][ig][ip][it],
04885                  in);
04886         }
04887     }
04888 }
04889
04890 /* Error message... */
04891 else
04892     ERRMSG("Unknown look-up table format!");
04893
04894 /* Close file... */
04895 fclose(in);
04896 }
04897 }

```

**5.19.3.43 refractivity()** double refractivity (  
double p,  
double t )

Compute refractivity (return value is n - 1).

Definition at line 4901 of file [jurassic.c](#).

```

04903 {
04904
04905     /* Refractivity of air at 4 to 15 micron... */
04906     return 7.753e-05 * p / t;
04907 }

```

**5.19.3.44 scan\_ctl()** double scan\_ctl (  
int argc,  
char \* argv[],  
const char \* varname,

```

    int arridx,
    const char * defvalue,
    char * value )

```

Search control parameter file for variable entry.

Definition at line 4911 of file [jurassic.c](#).

```

04917     {
04918
04919     FILE *in = NULL;
04920
04921     char dummy[LEN], fullname1[LEN], fullname2[LEN], line[LEN],
04922         rvarname[LEN], rval[LEN];
04923
04924     int contain = 0;
04925
04926     /* Open file... */
04927     if (argv[1][0] != '-')
04928         if (!(in = fopen(argv[1], "r")))
04929             ERRMSG("Cannot open file!");
04930
04931     /* Set full variable name... */
04932     if (arridx >= 0) {
04933         sprintf(fullname1, "%s[%d]", varname, arridx);
04934         sprintf(fullname2, "%s[*]", varname);
04935     } else {
04936         sprintf(fullname1, "%s", varname);
04937         sprintf(fullname2, "%s", varname);
04938     }
04939
04940     /* Read data... */
04941     if (in != NULL)
04942         while (fgets(line, LEN, in))
04943             if (sscanf(line, "%s %s %s", rvarname, dummy, rval) == 3)
04944                 if (strcasecmp(rvarname, fullname1) == 0 ||
04945                     strcasecmp(rvarname, fullname2) == 0) {
04946                     contain = 1;
04947                     break;
04948                 }
04949     for (int i = 1; i < argc - 1; i++)
04950         if (strcasecmp(argv[i], fullname1) == 0 ||
04951             strcasecmp(argv[i], fullname2) == 0) {
04952             sprintf(rval, "%s", argv[i + 1]);
04953             contain = 1;
04954             break;
04955         }
04956
04957     /* Close file... */
04958     if (in != NULL)
04959         fclose(in);
04960
04961     /* Check for missing variables... */
04962     if (!contain) {
04963         if (strlen(defvalue) > 0)
04964             sprintf(rval, "%s", defvalue);
04965         else
04966             ERRMSG("Missing variable %s!\n", fullname1);
04967     }
04968
04969     /* Write info... */
04970     LOG(1, "%s = %s", fullname1, rval);
04971
04972     /* Return values... */
04973     if (value != NULL)
04974         sprintf(value, "%s", rval);
04975     return atof(rval);
04976 }

```

**5.19.3.45** **sza()** double sza (  
     double sec,  
     double lon,  
     double lat )

Calculate solar zenith angle.

Definition at line 4980 of file [jurassic.c](#).

```

04983         {
04984
04985         /* Number of days and fraction with respect to 2000-01-01T12:00Z... */
04986         double D = sec / 86400 - 0.5;
04987
04988         /* Geocentric apparent ecliptic longitude [rad]... */
04989         double g = (357.529 + 0.98560028 * D) * M_PI / 180;
04990         double q = 280.459 + 0.98564736 * D;
04991         double L = (q + 1.915 * sin(g) + 0.020 * sin(2 * g)) * M_PI / 180;
04992
04993         /* Mean obliquity of the ecliptic [rad]... */
04994         double e = (23.439 - 0.00000036 * D) * M_PI / 180;
04995
04996         /* Declination [rad]... */
04997         double dec = asin(sin(e) * sin(L));
04998
04999         /* Right ascension [rad]... */
05000         double ra = atan2(cos(e) * sin(L), cos(L));
05001
05002         /* Greenwich Mean Sidereal Time [h]... */
05003         double GMST = 18.697374558 + 24.06570982441908 * D;
05004
05005         /* Local Sidereal Time [h]... */
05006         double LST = GMST + lon / 15;
05007
05008         /* Hour angle [rad]... */
05009         double h = LST / 12 * M_PI - ra;
05010
05011         /* Convert latitude... */
05012         lat *= M_PI / 180;
05013
05014         /* Return solar zenith angle [deg]... */
05015         return acos(sin(lat) * sin(dec) +
05016                   cos(lat) * cos(dec) * cos(h)) * 180 / M_PI;
05017 }

```

**5.19.3.46 tangent\_point()** void tangent\_point (

```

    los_t * los,
    double * tpz,
    double * tplon,
    double * tplat )

```

Find tangent point of a given LOS.

Definition at line 5021 of file [jurassic.c](#).

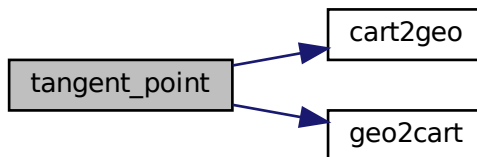
```

05025         {
05026
05027         double dummy, v[3], v0[3], v2[3];
05028
05029         /* Find minimum altitude... */
05030         size_t ip = gsl_stats_min_index(los->z, 1, (size_t) los->np);
05031
05032         /* Nadir or zenith... */
05033         if (ip <= 0 || ip >= (size_t) los->np - 1) {
05034             *tpz = los->z[los->np - 1];
05035             *tplon = los->lon[los->np - 1];
05036             *tplat = los->lat[los->np - 1];
05037         }
05038
05039         /* Limb... */
05040         else {
05041
05042             /* Determine interpolating polynomial y=a*x^2+b*x+c... */
05043             double yy0 = los->z[ip - 1];
05044             double yy1 = los->z[ip];
05045             double yy2 = los->z[ip + 1];
05046             double x1 = sqrt(POW2(los->ds[ip]) - POW2(yy1 - yy0));
05047             double x2 = x1 + sqrt(POW2(los->ds[ip + 1]) - POW2(yy2 - yy1));
05048             double a = 1 / (x1 - x2) * (-(yy0 - yy1) / x1 + (yy0 - yy2) / x2);
05049             double b = -(yy0 - yy1) / x1 - a * x1;
05050             double c = yy0;
05051
05052             /* Get tangent point location... */
05053             double x = -b / (2 * a);
05054             *tpz = a * x * x + b * x + c;
05055             geo2cart(los->z[ip - 1], los->lon[ip - 1], los->lat[ip - 1], v0);
05056             geo2cart(los->z[ip + 1], los->lon[ip + 1], los->lat[ip + 1], v2);

```

```
05057     for (int i = 0; i < 3; i++)
05058         v[i] = LIN(0.0, v0[i], x2, v2[i], x);
05059     cart2geo(v, &dummy, tplon, tplat);
05060 }
05061 }
```

Here is the call graph for this function:



```
5.19.3.47 time2jsec() void time2jsec (
    int year,
    int mon,
    int day,
    int hour,
    int min,
    int sec,
    double remain,
    double * jsec )
```

Convert date to seconds.

Definition at line 5065 of file [jurassic.c](#).

```
05073     {
05074
05075     struct tm t0, t1;
05076
05077     t0.tm_year = 100;
05078     t0.tm_mon = 0;
05079     t0.tm_mday = 1;
05080     t0.tm_hour = 0;
05081     t0.tm_min = 0;
05082     t0.tm_sec = 0;
05083
05084     t1.tm_year = year - 1900;
05085     t1.tm_mon = mon - 1;
05086     t1.tm_mday = day;
05087     t1.tm_hour = hour;
05088     t1.tm_min = min;
05089     t1.tm_sec = sec;
05090
05091     *jsec = (double) timegm(&t1) - (double) timegm(&t0) + remain;
05092 }
```

```

5.19.3.48 timer() void timer (
    const char * name,
    const char * file,
    const char * func,
    int line,
    int mode )

```

Measure wall-clock time.

Definition at line 5096 of file [jurassic.c](#).

```

05101     {
05102
05103     static double w0[10];
05104
05105     static int l0[10], nt;
05106
05107     /* Start new timer... */
05108     if (mode == 1) {
05109         w0[nt] = omp_get_wtime();
05110         l0[nt] = line;
05111         if ((++nt) >= 10)
05112             ERRMSG("Too many timers!");
05113     }
05114
05115     /* Write elapsed time... */
05116     else {
05117
05118         /* Check timer index... */
05119         if (nt - 1 < 0)
05120             ERRMSG("Coding error!");
05121
05122         /* Write elapsed time... */
05123         LOG(1, "Timer '%s' (%s, %s, l%d-%d): %.3f sec",
05124             name, file, func, l0[nt - 1], line, omp_get_wtime() - w0[nt - 1]);
05125     }
05126
05127     /* Stop timer... */
05128     if (mode == 3)
05129         nt--;
05130 }

```

```

5.19.3.49 write_atm() void write_atm (
    const char * dirname,
    const char * filename,
    ctl_t * ctl,
    atm_t * atm )

```

Write atmospheric data.

Definition at line 5134 of file [jurassic.c](#).

```

05138     {
05139
05140     FILE *out;
05141
05142     char file[LEN];
05143
05144     int n = 6;
05145
05146     /* Set filename... */
05147     if (dirname != NULL)
05148         sprintf(file, "%s/%s", dirname, filename);
05149     else
05150         sprintf(file, "%s", filename);
05151
05152     /* Write info... */
05153     LOG(1, "Write atmospheric data: %s", file);
05154
05155     /* Create file... */
05156     if (!(out = fopen(file, "w")))
05157         ERRMSG("Cannot create file!");
05158
05159     /* Write header... */
05160     fprintf(out,

```

```

05161         "# $1 = time (seconds since 2000-01-01T00:00Z)\n"
05162         "# $2 = altitude [km]\n"
05163         "# $3 = longitude [deg]\n"
05164         "# $4 = latitude [deg]\n"
05165         "# $5 = pressure [hPa]\n" "# $6 = temperature [K]\n");
05166     for (int ig = 0; ig < ctl->ng; ig++)
05167         fprintf(out, "# $d = %s volume mixing ratio [ppv]\n",
05168             ++n, ctl->emitter[ig]);
05169     for (int iw = 0; iw < ctl->nw; iw++)
05170         fprintf(out, "# $d = extinction (window %d) [1/km]\n", ++n, iw);
05171     if (ctl->ncl > 0) {
05172         fprintf(out, "# $d = cloud layer height [km]\n", ++n);
05173         fprintf(out, "# $d = cloud layer depth [km]\n", ++n);
05174         for (int icl = 0; icl < ctl->ncl; icl++)
05175             fprintf(out, "# $d = cloud layer extinction (%.4f cm^-1) [1/km]\n",
05176                 ++n, ctl->clnu[icl]);
05177     }
05178     if (ctl->nsf > 0) {
05179         fprintf(out, "# $d = surface layer height [km]\n", ++n);
05180         fprintf(out, "# $d = surface layer pressure [hPa]\n", ++n);
05181         fprintf(out, "# $d = surface layer temperature [K]\n", ++n);
05182         for (int isf = 0; isf < ctl->nsf; isf++)
05183             fprintf(out, "# $d = surface layer emissivity (%.4f cm^-1)\n",
05184                 ++n, ctl->sfnu[isf]);
05185     }
05186     /* Write data... */
05187     for (int ip = 0; ip < atm->np; ip++) {
05188         if (ip == 0 || atm->time[ip] != atm->time[ip - 1])
05189             fprintf(out, "\n");
05190         fprintf(out, "%.2f %g %g %g %g %g", atm->time[ip], atm->z[ip],
05191             atm->lon[ip], atm->lat[ip], atm->p[ip], atm->t[ip]);
05192         for (int ig = 0; ig < ctl->ng; ig++)
05193             fprintf(out, " %g", atm->q[ig][ip]);
05194         for (int iw = 0; iw < ctl->nw; iw++)
05195             fprintf(out, " %g", atm->k[iw][ip]);
05196         if (ctl->ncl > 0) {
05197             fprintf(out, " %g %g", atm->clz, atm->cldz);
05198             for (int icl = 0; icl < ctl->ncl; icl++)
05199                 fprintf(out, " %g", atm->clk[icl]);
05200         }
05201         if (ctl->nsf > 0) {
05202             fprintf(out, " %g %g %g", atm->sfz, atm->sfp, atm->sft);
05203             for (int isf = 0; isf < ctl->nsf; isf++)
05204                 fprintf(out, " %g", atm->sfeps[isf]);
05205         }
05206         fprintf(out, "\n");
05207     }
05208     /* Close file... */
05209     fclose(out);
05210 }

```

**5.19.3.50 write\_atm\_rfm()** void write\_atm\_rfm (  
const char \* filename,  
ctl\_t \* ctl,  
atm\_t \* atm )

Write atmospheric data in RFM format.

Definition at line 5216 of file [jurassic.c](#).

```

05219     {
05220     FILE *out;
05221     int ig, ip;
05222     /* Write info... */
05223     printf("Write RFM data: %s\n", filename);
05224     /* Create file... */
05225     if (!(out = fopen(filename, "w")))
05226         ERRMSG("Cannot create file!");
05227     /* Write data... */
05228     fprintf(out, "%d\n", atm->np);
05229     fprintf(out, "%HGT [km]\n");
05230     for (ip = 0; ip < atm->np; ip++)

```

```

05236     fprintf(out, "%g\n", atm->z[ip]);
05237     fprintf(out, "*PRE [mb]\n");
05238     for (ip = 0; ip < atm->np; ip++)
05239         fprintf(out, "%g\n", atm->p[ip]);
05240     fprintf(out, "*TEM [K]\n");
05241     for (ip = 0; ip < atm->np; ip++)
05242         fprintf(out, "%g\n", atm->t[ip]);
05243     for (ig = 0; ig < ctl->ng; ig++) {
05244         fprintf(out, "%s [ppmv]\n", ctl->emitter[ig]);
05245         for (ip = 0; ip < atm->np; ip++)
05246             fprintf(out, "%g\n", atm->q[ig][ip] * 1e6);
05247     }
05248     fprintf(out, "END\n");
05249
05250     /* Close file... */
05251     fclose(out);
05252 }

```

#### 5.19.3.51 write\_matrix() void write\_matrix (

```

    const char * dirname,
    const char * filename,
    ctl_t * ctl,
    gsl_matrix * matrix,
    atm_t * atm,
    obs_t * obs,
    const char * rowspace,
    const char * colspace,
    const char * sort )

```

Write matrix.

Definition at line 5256 of file jurassic.c.

```

05265     {
05266
05267     FILE *out;
05268
05269     char file[LEN], quantity[LEN];
05270
05271     int *cida, *ciqa, *cipa, *cira, *rida, *riqa, *ripa, *rira;
05272
05273     size_t i, j, nc, nr;
05274
05275     /* Check output flag... */
05276     if (!ctl->write_matrix)
05277         return;
05278
05279     /* Allocate... */
05280     ALLOC(cida, int,
05281           M);
05282     ALLOC(ciqa, int,
05283           N);
05284     ALLOC(cipa, int,
05285           N);
05286     ALLOC(cira, int,
05287           M);
05288     ALLOC(rida, int,
05289           M);
05290     ALLOC(riqa, int,
05291           N);
05292     ALLOC(ripa, int,
05293           N);
05294     ALLOC(rira, int,
05295           M);
05296
05297     /* Set filename... */
05298     if (dirname != NULL)
05299         sprintf(file, "%s/%s", dirname, filename);
05300     else
05301         sprintf(file, "%s", filename);
05302
05303     /* Write info... */
05304     LOG(1, "Write matrix: %s", file);
05305
05306     /* Create file... */
05307     if (!(out = fopen(file, "w")))

```

```

05308     ERRMSG("Cannot create file!");
05309
05310     /* Write header (row space)... */
05311     if (rowSpace[0] == 'y') {
05312
05313         fprintf(out,
05314             "# $1 = Row: index (measurement space)\n"
05315             "# $2 = Row: channel wavenumber [cm^-1]\n"
05316             "# $3 = Row: time (seconds since 2000-01-01T00:00Z)\n"
05317             "# $4 = Row: view point altitude [km]\n"
05318             "# $5 = Row: view point longitude [deg]\n"
05319             "# $6 = Row: view point latitude [deg]\n");
05320
05321         /* Get number of rows... */
05322         nr = obs2y(ctl, obs, NULL, rida, rira);
05323
05324     } else {
05325
05326         fprintf(out,
05327             "# $1 = Row: index (state space)\n"
05328             "# $2 = Row: name of quantity\n"
05329             "# $3 = Row: time (seconds since 2000-01-01T00:00Z)\n"
05330             "# $4 = Row: altitude [km]\n"
05331             "# $5 = Row: longitude [deg]\n" "# $6 = Row: latitude [deg]\n");
05332
05333         /* Get number of rows... */
05334         nr = atm2x(ctl, atm, NULL, rira, ripa);
05335     }
05336
05337     /* Write header (column space)... */
05338     if (colSpace[0] == 'y') {
05339
05340         fprintf(out,
05341             "# $7 = Col: index (measurement space)\n"
05342             "# $8 = Col: channel wavenumber [cm^-1]\n"
05343             "# $9 = Col: time (seconds since 2000-01-01T00:00Z)\n"
05344             "# $10 = Col: view point altitude [km]\n"
05345             "# $11 = Col: view point longitude [deg]\n"
05346             "# $12 = Col: view point latitude [deg]\n");
05347
05348         /* Get number of columns... */
05349         nc = obs2y(ctl, obs, NULL, cida, cira);
05350
05351     } else {
05352
05353         fprintf(out,
05354             "# $7 = Col: index (state space)\n"
05355             "# $8 = Col: name of quantity\n"
05356             "# $9 = Col: time (seconds since 2000-01-01T00:00Z)\n"
05357             "# $10 = Col: altitude [km]\n"
05358             "# $11 = Col: longitude [deg]\n" "# $12 = Col: latitude [deg]\n");
05359
05360         /* Get number of columns... */
05361         nc = atm2x(ctl, atm, NULL, cira, cipa);
05362     }
05363
05364     /* Write header entry... */
05365     fprintf(out, "# $13 = Matrix element\n\n");
05366
05367     /* Write matrix data... */
05368     i = j = 0;
05369     while (i < nr && j < nc) {
05370
05371         /* Write info about the row... */
05372         if (rowSpace[0] == 'y')
05373             fprintf(out, "%d %.4f %.2f %g %g %g",
05374                 (int) i, ctl->nu[rida[i]],
05375                 obs->time[rira[i]], obs->vpz[rira[i]],
05376                 obs->vplon[rira[i]], obs->vplat[rira[i]]);
05377         else {
05378             idx2name(ctl, rira[i], quantity);
05379             fprintf(out, "%d %s %.2f %g %g %g", (int) i, quantity,
05380                 atm->time[ripa[i]], atm->z[ripa[i]],
05381                 atm->lon[ripa[i]], atm->lat[ripa[i]]);
05382         }
05383
05384         /* Write info about the column... */
05385         if (colSpace[0] == 'y')
05386             fprintf(out, " %d %.4f %.2f %g %g %g",
05387                 (int) j, ctl->nu[cida[j]],
05388                 obs->time[cira[j]], obs->vpz[cira[j]],
05389                 obs->vplon[cira[j]], obs->vplat[cira[j]]);
05390         else {
05391             idx2name(ctl, cira[j], quantity);
05392             fprintf(out, " %d %s %.2f %g %g %g", (int) j, quantity,
05393                 atm->time[cipa[j]], atm->z[cipa[j]],
05394                 atm->lon[cipa[j]], atm->lat[cipa[j]]);

```

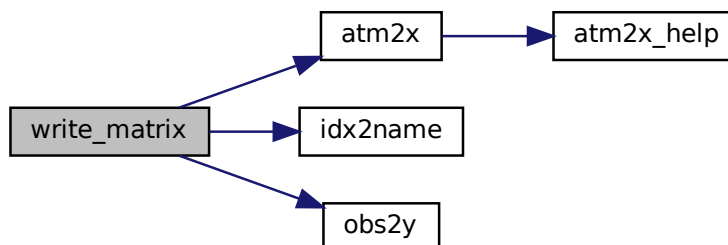


```

05395     }
05396
05397     /* Write matrix entry... */
05398     fprintf(out, " %g\n", gsl_matrix_get(matrix, i, j));
05399
05400     /* Set matrix indices... */
05401     if (sort[0] == 'r') {
05402         j++;
05403         if (j >= nc) {
05404             j = 0;
05405             i++;
05406             fprintf(out, "\n");
05407         }
05408     } else {
05409         i++;
05410         if (i >= nr) {
05411             i = 0;
05412             j++;
05413             fprintf(out, "\n");
05414         }
05415     }
05416 }
05417
05418 /* Close file... */
05419 fclose(out);
05420
05421 /* Free... */
05422 free(cida);
05423 free(ciga);
05424 free(cipa);
05425 free(cira);
05426 free(rida);
05427 free(riqa);
05428 free(ripa);
05429 free(rira);
05430 }

```

Here is the call graph for this function:



**5.19.3.52 write\_obs()** void write\_obs (

```

    const char * dirname,
    const char * filename,
    ctl_t * ctl,
    obs_t * obs )

```

Write observation data.

Definition at line 5434 of file [jurassic.c](#).

```

05438     {
05439
05440     FILE *out;

```

```

05441
05442     char file[LEN];
05443
05444     int n = 10;
05445
05446     /* Set filename... */
05447     if (dirname != NULL)
05448         sprintf(file, "%s/%s", dirname, filename);
05449     else
05450         sprintf(file, "%s", filename);
05451
05452     /* Write info... */
05453     LOG(1, "Write observation data: %s", file);
05454
05455     /* Create file... */
05456     if (!(out = fopen(file, "w")))
05457         ERRMSG("Cannot create file!");
05458
05459     /* Write header... */
05460     fprintf(out,
05461             "# $1 = time (seconds since 2000-01-01T00:00Z)\n"
05462             "# $2 = observer altitude [km]\n"
05463             "# $3 = observer longitude [deg]\n"
05464             "# $4 = observer latitude [deg]\n"
05465             "# $5 = view point altitude [km]\n"
05466             "# $6 = view point longitude [deg]\n"
05467             "# $7 = view point latitude [deg]\n"
05468             "# $8 = tangent point altitude [km]\n"
05469             "# $9 = tangent point longitude [deg]\n"
05470             "# $10 = tangent point latitude [deg]\n");
05471     for (int id = 0; id < ctl->nd; id++)
05472         fprintf(out, "# $%d = radiance (%.4f cm^-1) [W/(m^2 sr cm^-1)]\n",
05473             ++n, ctl->nu[id]);
05474     for (int id = 0; id < ctl->nd; id++)
05475         fprintf(out, "# $%d = transmittance (%.4f cm^-1) [-]\n", ++n,
05476             ctl->nu[id]);
05477
05478     /* Write data... */
05479     for (int ir = 0; ir < obs->nr; ir++) {
05480         if (ir == 0 || obs->time[ir] != obs->time[ir - 1])
05481             fprintf(out, "\n");
05482         fprintf(out, "%.2f %g %g %g %g %g %g %g %g", obs->time[ir],
05483             obs->obsz[ir], obs->obslon[ir], obs->obslat[ir],
05484             obs->vpz[ir], obs->vplon[ir], obs->vplat[ir],
05485             obs->tpz[ir], obs->tplon[ir], obs->tplat[ir]);
05486         for (int id = 0; id < ctl->nd; id++)
05487             fprintf(out, " %g", obs->rad[id][ir]);
05488         for (int id = 0; id < ctl->nd; id++)
05489             fprintf(out, " %g", obs->tau[id][ir]);
05490         fprintf(out, "\n");
05491     }
05492
05493     /* Close file... */
05494     fclose(out);
05495 }

```

**5.19.3.53 write\_shape()** void write\_shape (  
     const char \* filename,  
     double \* x,  
     double \* y,  
     int n )

Write shape function.

Definition at line 5499 of file [jurassic.c](#).

```

05503     {
05504
05505     FILE *out;
05506
05507     /* Write info... */
05508     LOG(1, "Write shape function: %s", filename);
05509
05510     /* Create file... */
05511     if (!(out = fopen(filename, "w")))
05512         ERRMSG("Cannot create file!");
05513
05514     /* Write header... */

```

```

05515     fprintf(out,
05516             "# $1 = shape function x-value [-]\n"
05517             "# $2 = shape function y-value [-]\n\n");
05518
05519     /* Write data... */
05520     for (int i = 0; i < n; i++)
05521         fprintf(out, "%.10g %.10g\n", x[i], y[i]);
05522
05523     /* Close file... */
05524     fclose(out);
05525 }

```

**5.19.3.54 write\_tbl()** void write\_tbl (

```

    ctl_t * ctl,
    tbl_t * tbl )

```

Write look-up table data.

Definition at line 5529 of file [jurassic.c](#).

```

05531     {
05532
05533     FILE *out;
05534
05535     char filename[2 * LEN];
05536
05537     /* Loop over emitters and detectors... */
05538     for (int ig = 0; ig < ctl->ng; ig++)
05539         for (int id = 0; id < ctl->nd; id++) {
05540
05541             /* Set filename... */
05542             sprintf(filename, "%s_%.4f_%.s.s", ctl->tblbase,
05543                     ctl->nu[id], ctl->emitter[id],
05544                     ctl->tblfmt == 1 ? "tab" : "bin");
05545
05546             /* Write info... */
05547             LOG(1, "Write emissivity table: %s", filename);
05548
05549             /* Create file... */
05550             if (!(out = fopen(filename, "w")))
05551                 ERRMSG("Cannot create file!");
05552
05553             /* Write ASCII data... */
05554             if (ctl->tblfmt == 1) {
05555
05556                 /* Write header... */
05557                 fprintf(out,
05558                         "# $1 = pressure [hPa]\n"
05559                         "# $2 = temperature [K]\n"
05560                         "# $3 = column density [molecules/cm^2]\n"
05561                         "# $4 = emissivity [-]\n");
05562
05563                 /* Save table file... */
05564                 for (int ip = 0; ip < tbl->np[id][ig]; ip++)
05565                     for (int it = 0; it < tbl->nt[id][ig][ip]; it++) {
05566                         fprintf(out, "\n");
05567                         for (int iu = 0; iu < tbl->nu[id][ig][ip][it]; iu++)
05568                             fprintf(out, "%g %g %e %e\n",
05569                                     tbl->p[id][ig][ip], tbl->t[id][ig][ip][it],
05570                                     tbl->u[id][ig][ip][it][iu],
05571                                     tbl->eps[id][ig][ip][it][iu]);
05572                     }
05573             }
05574
05575             /* Write binary data... */
05576             else if (ctl->tblfmt == 2) {
05577                 FWRITE(&tbl->np[id][ig], int,
05578                        1,
05579                        out);
05580                 FWRITE(tbl->p[id][ig], double,
05581                        (size_t) tbl->np[id][ig],
05582                        out);
05583                 for (int ip = 0; ip < tbl->np[id][ig]; ip++) {
05584                     FWRITE(&tbl->nt[id][ig][ip], int,
05585                            1,
05586                            out);
05587                     FWRITE(tbl->t[id][ig][ip], double,
05588                            (size_t) tbl->nt[id][ig][ip],
05589                            out);

```

```

05590         for (int it = 0; it < tbl->nt[id][ig][ip]; it++) {
05591             FWRITE(&tbl->nu[id][ig][ip][it], int,
05592                 1,
05593                 out);
05594             FWRITE(tbl->u[id][ig][ip][it], float,
05595                 (size_t) tbl->nu[id][ig][ip][it],
05596                 out);
05597             FWRITE(tbl->eps[id][ig][ip][it], float,
05598                 (size_t) tbl->nu[id][ig][ip][it],
05599                 out);
05600         }
05601     }
05602 }
05603
05604 /* Error message... */
05605 else
05606     ERRMSG("Unknown look-up table format!");
05607
05608 /* Close file... */
05609 fclose(out);
05610 }
05611 }

```

**5.19.3.55 x2atm()** void x2atm (

```

    ctl_t * ctl,
    gsl_vector * x,
    atm_t * atm )

```

Decompose parameter vector or state vector.

Definition at line 5615 of file [jurassic.c](#).

```

05618     {
05619
05620         size_t n = 0;
05621
05622         /* Get pressure... */
05623         for (int ip = 0; ip < atm->np; ip++)
05624             if (atm->z[ip] >= ctl->retp_zmin && atm->z[ip] <= ctl->retp_zmax)
05625                 x2atm_help(&atm->p[ip], x, &n);
05626
05627         /* Get temperature... */
05628         for (int ip = 0; ip < atm->np; ip++)
05629             if (atm->z[ip] >= ctl->rett_zmin && atm->z[ip] <= ctl->rett_zmax)
05630                 x2atm_help(&atm->t[ip], x, &n);
05631
05632         /* Get volume mixing ratio... */
05633         for (int ig = 0; ig < ctl->ng; ig++)
05634             for (int ip = 0; ip < atm->np; ip++)
05635                 if (atm->z[ip] >= ctl->retq_zmin[ig]
05636                     && atm->z[ip] <= ctl->retq_zmax[ig])
05637                     x2atm_help(&atm->q[ig][ip], x, &n);
05638
05639         /* Get extinction... */
05640         for (int iw = 0; iw < ctl->nw; iw++)
05641             for (int ip = 0; ip < atm->np; ip++)
05642                 if (atm->z[ip] >= ctl->retk_zmin[iw]
05643                     && atm->z[ip] <= ctl->retk_zmax[iw])
05644                     x2atm_help(&atm->k[iw][ip], x, &n);
05645
05646         /* Get cloud data... */
05647         if (ctl->ret_clz)
05648             x2atm_help(&atm->clz, x, &n);
05649         if (ctl->ret_cldz)
05650             x2atm_help(&atm->cldz, x, &n);
05651         if (ctl->ret_clk)
05652             for (int icl = 0; icl < ctl->ncl; icl++)
05653                 x2atm_help(&atm->clk[icl], x, &n);
05654
05655         /* Get surface data... */
05656         if (ctl->ret_sfz)
05657             x2atm_help(&atm->sfz, x, &n);
05658         if (ctl->ret_sfp)
05659             x2atm_help(&atm->sfp, x, &n);
05660         if (ctl->ret_sft)
05661             x2atm_help(&atm->sft, x, &n);
05662         if (ctl->ret_sfeps)
05663             for (int isf = 0; isf < ctl->nsf; isf++)
05664                 x2atm_help(&atm->sfeps[isf], x, &n);

```

```
05665 }
```

Here is the call graph for this function:



**5.19.3.56 x2atm\_help()** void x2atm\_help (

```

    double * value,
    gsl_vector * x,
    size_t * n )
```

Get element from state vector.

Definition at line [5669](#) of file [jurassic.c](#).

```

05672     {
05673
05674     /* Get state vector element... */
05675     *value = gsl_vector_get(x, *n);
05676     (*n)++;
05677 }
```

**5.19.3.57 y2obs()** void y2obs (

```

    ctl_t * ctl,
    gsl_vector * y,
    obs_t * obs )
```

Decompose measurement vector.

Definition at line [5681](#) of file [jurassic.c](#).

```

05684     {
05685
05686     size_t m = 0;
05687
05688     /* Decompose measurement vector... */
05689     for (int ir = 0; ir < obs->nr; ir++)
05690         for (int id = 0; id < ctl->nd; id++)
05691             if (gsl_finite(obs->rad[id][ir])) {
05692                 obs->rad[id][ir] = gsl_vector_get(y, m);
05693                 m++;
05694             }
05695 }
```

## 5.20 jurassic.h

```

00001 /*
00002   This file is part of JURASSIC.
00003
00004   JURASSIC is free software: you can redistribute it and/or modify
00005   it under the terms of the GNU General Public License as published by
00006   the Free Software Foundation, either version 3 of the License, or
00007   (at your option) any later version.
00008
00009   JURASSIC is distributed in the hope that it will be useful,
00010   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012   GNU General Public License for more details.
00013
00014   You should have received a copy of the GNU General Public License
00015   along with JURASSIC. If not, see <http://www.gnu.org/licenses/>.
00016
00017   Copyright (C) 2003-2021 Forschungszentrum Juelich GmbH
00018 */
00019
00037 #ifndef JURASSIC_H
00038 #define JURASSIC_H
00039
00040 #include <gsl/gsl_math.h>
00041 #include <gsl/gsl_blas.h>
00042 #include <gsl/gsl_linalg.h>
00043 #include <gsl/gsl_randist.h>
00044 #include <gsl/gsl_rng.h>
00045 #include <gsl/gsl_statistics.h>
00046 #include <math.h>
00047 #include <omp.h>
00048 #include <stdio.h>
00049 #include <stdlib.h>
00050 #include <string.h>
00051 #include <time.h>
00052
00053 /* -----
00054   Macros...
00055   ----- */
00056
00058 #define ALLOC(ptr, type, n) \
00059   if((ptr=malloc((size_t)(n)*sizeof(type)))==NULL) \
00060     ERRMSG("Out of memory!");
00061
00063 #define DIST(a, b) sqrt(DIST2(a, b))
00064
00066 #define DIST2(a, b) \
00067   ((a[0]-b[0])*(a[0]-b[0])+(a[1]-b[1])*(a[1]-b[1])+(a[2]-b[2])*(a[2]-b[2]))
00068
00070 #define DOTP(a, b) (a[0]*b[0]+a[1]*b[1]+a[2]*b[2])
00071
00073 #define EXP(x0, y0, x1, y1, x) \
00074   (((y0)>0 && (y1)>0) \
00075    ? ((y0)*exp(log((y1)/(y0))/((x1)-(x0))*((x)-(x0)))) \
00076    : LIN(x0, y0, x1, y1, x))
00077
00079 #define FREAD(ptr, type, size, out) { \
00080   if(fread(ptr, sizeof(type), size, out)!=size) \
00081     ERRMSG("Error while reading!"); \
00082 }
00083
00085 #define FWRITE(ptr, type, size, out) { \
00086   if(fwrite(ptr, sizeof(type), size, out)!=size) \
00087     ERRMSG("Error while writing!"); \
00088 }
00089
00091 #define LIN(x0, y0, x1, y1, x) \
00092   ((y0)+((y1)-(y0))/((x1)-(x0))*((x)-(x0)))
00093
00095 #define NORM(a) sqrt(DOTP(a, a))
00096
00098 #define POW2(x) ((x)*(x))
00099
00101 #define POW3(x) ((x)*(x)*(x))
00102
00104 #define TIMER(name, mode) \
00105   {timer(name, __FILE__, __func__, __LINE__, mode);}
00106
00108 #define TOK(line, tok, format, var) { \
00109   if(((tok)=strtok((line), " \t"))) { \
00110     if(sscanf(tok, format, &(var))!=1) continue; \
00111   } else ERRMSG("Error while reading!"); \
00112 }
00113
00114 /* -----
00115   Log messages...

```

```

00116  ----- */
00117
00119 #ifndef LOGLEV
00120 #define LOGLEV 2
00121 #endif
00122
00124 #define LOG(level, ...) {
00125     if(level >= 2)
00126         printf(" ");
00127     if(level <= LOGLEV) {
00128         printf(__VA_ARGS__);
00129         printf("\n");
00130     }
00131 }
00132
00134 #define WARN(...) {
00135     printf("\nWarning (%s, %s, l%d): ", __FILE__, __func__, __LINE__);
00136     LOG(0, __VA_ARGS__);
00137 }
00138
00140 #define ERRMSG(...) {
00141     printf("\nError (%s, %s, l%d): ", __FILE__, __func__, __LINE__);
00142     LOG(0, __VA_ARGS__);
00143     exit(EXIT_FAILURE);
00144 }
00145
00147 #define PRINT(format, var)
00148     printf("Print (%s, %s, l%d): %s= "format"\n",
00149         __FILE__, __func__, __LINE__, #var, var);
00150
00151 /* -----
00152     Constants...
00153     ----- */
00154
00156 #define TMIN 100.
00157
00159 #define TMAX 400.
00160
00162 #define TSUN 5780.
00163
00165 #define C1 1.19104259e-8
00166
00168 #define C2 1.43877506
00169
00171 #define G0 9.80665
00172
00174 #define KB 1.3806504e-23
00175
00177 #define NA 6.02214199e23
00178
00180 #define H0 7.0
00181
00183 #define P0 1013.25
00184
00186 #define T0 273.15
00187
00189 #define RE 6367.421
00190
00192 #define RI 8.3144598
00193
00195 #define ME 5.976e24
00196
00197 /* -----
00198     Dimensions...
00199     ----- */
00200
00202 #ifndef NCL
00203 #define NCL 8
00204 #endif
00205
00207 #ifndef ND
00208 #define ND 64
00209 #endif
00210
00212 #ifndef NG
00213 #define NG 8
00214 #endif
00215
00217 #ifndef NP
00218 #define NP 256
00219 #endif
00220
00222 #ifndef NR
00223 #define NR 256
00224 #endif
00225
00227 #ifndef NSF

```

```

00228 #define NSF 8
00229 #endif
00230
00232 #ifndef NW
00233 #define NW 4
00234 #endif
00235
00237 #ifndef LEN
00238 #define LEN 10000
00239 #endif
00240
00242 #ifndef M
00243 #define M (NR*ND)
00244 #endif
00245
00247 #ifndef N
00248 #define N ((2+NG+NW)*NP+NCL+NSF+5)
00249 #endif
00250
00252 #ifndef NQ
00253 #define NQ (7+NG+NW+NCL+NSF)
00254 #endif
00255
00257 #ifndef NLOS
00258 #define NLOS 4096
00259 #endif
00260
00262 #ifndef NSHAPE
00263 #define NSHAPE 10000
00264 #endif
00265
00267 #ifndef NFOV
00268 #define NFOV 5
00269 #endif
00270
00272 #ifndef TBLNP
00273 #define TBLNP 41
00274 #endif
00275
00277 #ifndef TBLNT
00278 #define TBLNT 30
00279 #endif
00280
00282 #ifndef TBLNU
00283 #define TBLNU 320
00284 #endif
00285
00287 #ifndef TBLNS
00288 #define TBLNS 1200
00289 #endif
00290
00292 #ifndef RFMNPTS
00293 #define RFMNPTS 10000000
00294 #endif
00295
00297 #ifndef RFMLINE
00298 #define RFMLINE 100000
00299 #endif
00300
00301 /* -----
00302     Quantity indices...
00303     ----- */
00304
00306 #define IDXP 0
00307
00309 #define IDXT 1
00310
00312 #define IDXQ(ig) (2+ig)
00313
00315 #define ID XK(iw) (2+ctl->ng+iw)
00316
00318 #define IDXCLZ (2+ctl->ng+ctl->nw)
00319
00321 #define IDXCLDZ (3+ctl->ng+ctl->nw)
00322
00324 #define IDXCLK(icl) (4+ctl->ng+ctl->nw+icl)
00325
00327 #define IDXSFZ (4+ctl->ng+ctl->nw+ctl->ncl)
00328
00330 #define IDXSFP (5+ctl->ng+ctl->nw+ctl->ncl)
00331
00333 #define IDXSFT (6+ctl->ng+ctl->nw+ctl->ncl)
00334
00336 #define IXSFEPs(isf) (7+ctl->ng+ctl->nw+ctl->ncl+isf)
00337
00338 /* -----
00339     Structs...

```



```

00340 ----- */
00341
00343 typedef struct {
00344
00346     int np;
00347
00349     double time[NP];
00350
00352     double z[NP];
00353
00355     double lon[NP];
00356
00358     double lat[NP];
00359
00361     double p[NP];
00362
00364     double t[NP];
00365
00367     double q[NG][NP];
00368
00370     double k[NW][NP];
00371
00373     double clz;
00374
00376     double cldz;
00377
00379     double clk[NCL];
00380
00382     double sfz;
00383
00385     double sfp;
00386
00388     double sft;
00389
00391     double sfeps[NSF];
00392
00393 } atm_t;
00394
00396 typedef struct {
00397
00399     int ng;
00400
00402     char emitter[NG][LEN];
00403
00405     int nd;
00406
00408     double nu[ND];
00409
00411     int nw;
00412
00414     int window[ND];
00415
00417     int ncl;
00418
00420     double clnu[NCL];
00421
00423     int nsf;
00424
00426     double sfnu[NSF];
00427
00429     int sftype;
00430
00432     double sfsza;
00433
00435     char tblbase[LEN];
00436
00438     int tblfmt;
00439
00441     double hyd;
00442
00444     int ctm_co2;
00445
00447     int ctm_h2o;
00448
00450     int ctm_n2;
00451
00453     int ctm_o2;
00454
00456     int refrac;
00457
00459     double rayds;
00460
00462     double raydz;
00463
00465     char fov[LEN];
00466
00468     double retp_zmin;

```

```
00469
00471 double retp_zmax;
00472
00474 double rett_zmin;
00475
00477 double rett_zmax;
00478
00480 double retq_zmin[NG];
00481
00483 double retq_zmax[NG];
00484
00486 double retk_zmin[NW];
00487
00489 double retk_zmax[NW];
00490
00492 int ret_clz;
00493
00495 int ret_cldz;
00496
00498 int ret_clk;
00499
00501 int ret_sfz;
00502
00504 int ret_sfp;
00505
00507 int ret_sft;
00508
00510 int ret_sfeps;
00511
00513 int write_bbt;
00514
00516 int write_matrix;
00517
00519 int formod;
00520
00522 char rfmbin[LEN];
00523
00525 char rfmhit[LEN];
00526
00528 char rfmxc[NG][LEN];
00529
00530 } ctl_t;
00531
00533 typedef struct {
00534
00536 int np;
00537
00539 double z[NLOS];
00540
00542 double lon[NLOS];
00543
00545 double lat[NLOS];
00546
00548 double p[NLOS];
00549
00551 double t[NLOS];
00552
00554 double q[NLOS][NG];
00555
00557 double k[NLOS][ND];
00558
00560 double sft;
00561
00563 double sfeps[ND];
00564
00566 double ds[NLOS];
00567
00569 double u[NLOS][NG];
00570
00572 double eps[NLOS][ND];
00573
00575 double src[NLOS][ND];
00576
00577 } los_t;
00578
00580 typedef struct {
00581
00583 int nr;
00584
00586 double time[NR];
00587
00589 double obsz[NR];
00590
00592 double obslon[NR];
00593
00595 double obslat[NR];
00596
```

```

00598 double vpz[NR];
00599
00601 double vplon[NR];
00602
00604 double vplat[NR];
00605
00607 double tpz[NR];
00608
00610 double tplon[NR];
00611
00613 double tplat[NR];
00614
00616 double tau[ND][NR];
00617
00619 double rad[ND][NR];
00620
00621 } obs_t;
00622
00624 typedef struct {
00625
00627 int np[ND][NG];
00628
00630 int nt[ND][NG][TBLNP];
00631
00633 int nu[ND][NG][TBLNP][TBLNT];
00634
00636 double p[ND][NG][TBLNP];
00637
00639 double t[ND][NG][TBLNP][TBLNT];
00640
00642 float u[ND][NG][TBLNP][TBLNT][TBLNU];
00643
00645 float eps[ND][NG][TBLNP][TBLNT][TBLNU];
00646
00648 double st[TBLNS];
00649
00651 double sr[TBLNS][ND];
00652
00653 } tbl_t;
00654
00655 /* -----
00656 Functions...
00657 ----- */
00658
00660 size_t atm2x(
00661     ctl_t * ctl,
00662     atm_t * atm,
00663     gsl_vector * x,
00664     int *iqa,
00665     int *ipa);
00666
00668 void atm2x_help(
00669     double value,
00670     int value_iqa,
00671     int value_ip,
00672     gsl_vector * x,
00673     int *iqa,
00674     int *ipa,
00675     size_t *n);
00676
00678 double brightness(
00679     double rad,
00680     double nu);
00681
00683 void cart2geo(
00684     double *x,
00685     double *z,
00686     double *lon,
00687     double *lat);
00688
00690 void climatology(
00691     ctl_t * ctl,
00692     atm_t * atm_mean);
00693
00695 double ctmc02(
00696     double nu,
00697     double p,
00698     double t,
00699     double u);
00700
00702 double ctmh2o(
00703     double nu,
00704     double p,
00705     double t,
00706     double q,
00707     double u);
00708

```

```
00710 double ctmn2(
00711     double nu,
00712     double p,
00713     double t);
00714
00716 double ctmo2(
00717     double nu,
00718     double p,
00719     double t);
00720
00722 void copy_atm(
00723     ctl_t * ctl,
00724     atm_t * atm_dest,
00725     atm_t * atm_src,
00726     int init);
00727
00729 void copy_obs(
00730     ctl_t * ctl,
00731     obs_t * obs_dest,
00732     obs_t * obs_src,
00733     int init);
00734
00736 int find_emitter(
00737     ctl_t * ctl,
00738     const char *emitter);
00739
00741 void formod(
00742     ctl_t * ctl,
00743     atm_t * atm,
00744     obs_t * obs);
00745
00747 void formod_continua(
00748     ctl_t * ctl,
00749     los_t * los,
00750     int ip,
00751     double *beta);
00752
00754 void formod_fov(
00755     ctl_t * ctl,
00756     obs_t * obs);
00757
00759 void formod_pencil(
00760     ctl_t * ctl,
00761     atm_t * atm,
00762     obs_t * obs,
00763     int ir);
00764
00766 void formod_rfm(
00767     ctl_t * ctl,
00768     atm_t * atm,
00769     obs_t * obs);
00770
00772 void formod_srcfunc(
00773     ctl_t * ctl,
00774     tbl_t * tbl,
00775     double t,
00776     double *src);
00777
00779 void geo2cart(
00780     double z,
00781     double lon,
00782     double lat,
00783     double *x);
00784
00786 void hydrostatic(
00787     ctl_t * ctl,
00788     atm_t * atm);
00789
00791 void idx2name(
00792     ctl_t * ctl,
00793     int idx,
00794     char *quantity);
00795
00797 void init_srcfunc(
00798     ctl_t * ctl,
00799     tbl_t * tbl);
00800
00802 void intpol_atm(
00803     ctl_t * ctl,
00804     atm_t * atm,
00805     double z,
00806     double *p,
00807     double *t,
00808     double *q,
00809     double *k);
00810
00812 void intpol_tbl(
```

```
00813     ctl_t * ctl,
00814     tbl_t * tbl,
00815     los_t * los,
00816     int ip,
00817     double tau_path[ND][NG],
00818     double tau_seg[ND]);
00819
00821 double intpol_tbl_eps(
00822     tbl_t * tbl,
00823     int ig,
00824     int id,
00825     int ip,
00826     int it,
00827     double u);
00828
00830 double intpol_tbl_u(
00831     tbl_t * tbl,
00832     int ig,
00833     int id,
00834     int ip,
00835     int it,
00836     double eps);
00837
00839 void jsec2time(
00840     double jsec,
00841     int *year,
00842     int *mon,
00843     int *day,
00844     int *hour,
00845     int *min,
00846     int *sec,
00847     double *remain);
00848
00850 void kernel(
00851     ctl_t * ctl,
00852     atm_t * atm,
00853     obs_t * obs,
00854     gsl_matrix * k);
00855
00857 int locate_irr(
00858     double *xx,
00859     int n,
00860     double x);
00861
00863 int locate_reg(
00864     double *xx,
00865     int n,
00866     double x);
00867
00869 int locate_tbl(
00870     float *xx,
00871     int n,
00872     double x);
00873
00875 size_t obs2y(
00876     ctl_t * ctl,
00877     obs_t * obs,
00878     gsl_vector * y,
00879     int *ida,
00880     int *ira);
00881
00883 double planck(
00884     double t,
00885     double nu);
00886
00888 void raytrace(
00889     ctl_t * ctl,
00890     atm_t * atm,
00891     obs_t * obs,
00892     los_t * los,
00893     int ir);
00894
00896 void read_atm(
00897     const char *dirname,
00898     const char *filename,
00899     ctl_t * ctl,
00900     atm_t * atm);
00901
00903 void read_ctl(
00904     int argc,
00905     char *argv[],
00906     ctl_t * ctl);
00907
00909 void read_matrix(
00910     const char *dirname,
00911     const char *filename,
00912     gsl_matrix * matrix);
```

```
00913
00915 void read_obs(
00916     const char *dirname,
00917     const char *filename,
00918     ctl_t * ctl,
00919     obs_t * obs);
00920
00922 double read_obs_rfm(
00923     const char *basename,
00924     double z,
00925     double *nu,
00926     double *f,
00927     int n);
00928
00930 void read_rfm_spec(
00931     const char *filename,
00932     double *nu,
00933     double *rad,
00934     int *npts);
00935
00937 void read_shape(
00938     const char *filename,
00939     double *x,
00940     double *y,
00941     int *n);
00942
00944 void read_tbl(
00945     ctl_t * ctl,
00946     tbl_t * tbl);
00947
00949 double refractivity(
00950     double p,
00951     double t);
00952
00954 double scan_ctl(
00955     int argc,
00956     char *argv[],
00957     const char *varname,
00958     int arridx,
00959     const char *defvalue,
00960     char *value);
00961
00963 double sza(
00964     double sec,
00965     double lon,
00966     double lat);
00967
00969 void tangent_point(
00970     los_t * los,
00971     double *tpz,
00972     double *tplon,
00973     double *tplat);
00974
00976 void time2jsec(
00977     int year,
00978     int mon,
00979     int day,
00980     int hour,
00981     int min,
00982     int sec,
00983     double remain,
00984     double *jsec);
00985
00987 void timer(
00988     const char *name,
00989     const char *file,
00990     const char *func,
00991     int line,
00992     int mode);
00993
00995 void write_atm(
00996     const char *dirname,
00997     const char *filename,
00998     ctl_t * ctl,
00999     atm_t * atm);
01000
01002 void write_atm_rfm(
01003     const char *filename,
01004     ctl_t * ctl,
01005     atm_t * atm);
01006
01008 void write_matrix(
01009     const char *dirname,
01010     const char *filename,
01011     ctl_t * ctl,
01012     gsl_matrix * matrix,
01013     atm_t * atm,
```

```
01014     obs_t * obs,  
01015     const char *rowspace,  
01016     const char *colspace,  
01017     const char *sort);  
01018  
01020 void write_obs(  
01021     const char *dirname,  
01022     const char *filename,  
01023     ctl_t * ctl,  
01024     obs_t * obs);  
01025  
01027 void write_shape(  
01028     const char *filename,  
01029     double *x,  
01030     double *y,  
01031     int n);  
01032  
01034 void write_tbl(  
01035     ctl_t * ctl,  
01036     tbl_t * tbl);  
01037  
01039 void x2atm(  
01040     ctl_t * ctl,  
01041     gsl_vector * x,  
01042     atm_t * atm);  
01043  
01045 void x2atm_help(  
01046     double *value,  
01047     gsl_vector * x,  
01048     size_t *n);  
01049  
01051 void y2obs(  
01052     ctl_t * ctl,  
01053     gsl_vector * y,  
01054     obs_t * obs);  
01055  
01056 #endif
```

## 5.21 kernel.c File Reference

Calculate kernel functions.

```
#include "jurassic.h"
```

### Functions

- int [main](#) (int argc, char \*argv[])

#### 5.21.1 Detailed Description

Calculate kernel functions.

Definition in file [kernel.c](#).

#### 5.21.2 Function Documentation

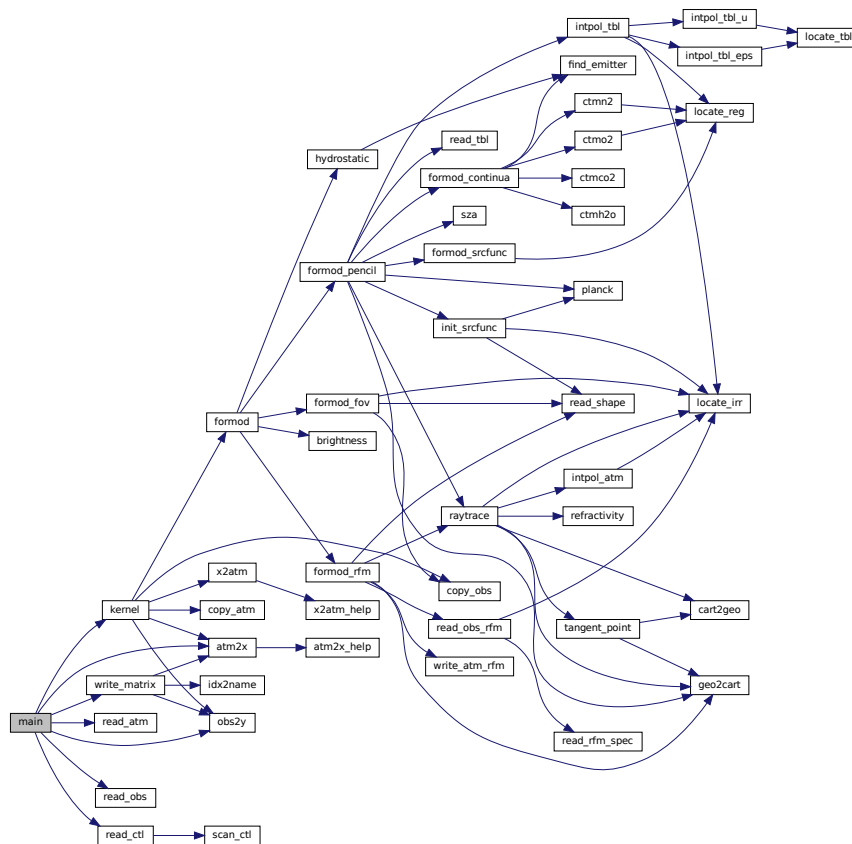
**5.21.2.1 main()** `int main (`  
    `int argc,`  
    `char * argv[] )`

Definition at line 27 of file [kernel.c](#).

```
00029     {
00030
00031     static atm_t atm;
00032     static ctl_t ctl;
00033     static obs_t obs;
00034
00035     gsl_matrix *k;
00036
00037     /* Check arguments... */
00038     if (argc < 5)
00039         ERRMSG("Give parameters: <ctl> <obs> <atm> <kernel>");
00040
00041     /* Read control parameters... */
00042     read_ctl(argc, argv, &ctl);
00043
00044     /* Set flags... */
00045     ctl.write_matrix = 1;
00046
00047     /* Read observation geometry... */
00048     read_obs(NULL, argv[2], &ctl, &obs);
00049
00050     /* Read atmospheric data... */
00051     read_atm(NULL, argv[3], &ctl, &atm);
00052
00053     /* Get sizes... */
00054     size_t n = atm2x(&ctl, &atm, NULL, NULL, NULL);
00055     size_t m = obs2y(&ctl, &obs, NULL, NULL, NULL);
00056
00057     /* Check sizes... */
00058     if (n == 0)
00059         ERRMSG("No state vector elements!");
00060     if (m == 0)
00061         ERRMSG("No measurement vector elements!");
00062
00063     /* Allocate... */
00064     k = gsl_matrix_alloc(m, n);
00065
00066     /* Compute kernel matrix... */
00067     kernel(&ctl, &atm, &obs, k);
00068
00069     /* Write matrix to file... */
00070     write_matrix(NULL, argv[4], &ctl, k, &atm, &obs, "y", "x", "r");
00071
00072     /* Free... */
00073     gsl_matrix_free(k);
00074
00075     return EXIT_SUCCESS;
00076 }
```



Here is the call graph for this function:



## 5.22 kernel.c

```

00001 /*
00002  This file is part of JURASSIC.
00003
00004  JURASSIC is free software: you can redistribute it and/or modify
00005  it under the terms of the GNU General Public License as published by
00006  the Free Software Foundation, either version 3 of the License, or
00007  (at your option) any later version.
00008
00009  JURASSIC is distributed in the hope that it will be useful,
00010  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  GNU General Public License for more details.
00013
00014  You should have received a copy of the GNU General Public License
00015  along with JURASSIC. If not, see <http://www.gnu.org/licenses/>.
00016
00017  Copyright (C) 2003-2021 Forschungszentrum Juelich GmbH
00018 */
00019
00025 #include "jurassic.h"
00026
00027 int main(
00028     int argc,
00029     char *argv[]) {
00030
00031     static atm_t atm;
00032     static ctl_t ctl;
00033     static obs_t obs;
00034
00035     gsl_matrix *k;
00036
00037     /* Check arguments... */
00038     if (argc < 5)
00039         ERRMSG("Give parameters: <ctl> <obs> <atm> <kernel>");

```

```

00040
00041  /* Read control parameters... */
00042  read_ctl(argc, argv, &ctl);
00043
00044  /* Set flags... */
00045  ctl.write_matrix = 1;
00046
00047  /* Read observation geometry... */
00048  read_obs(NULL, argv[2], &ctl, &obs);
00049
00050  /* Read atmospheric data... */
00051  read_atm(NULL, argv[3], &ctl, &atm);
00052
00053  /* Get sizes... */
00054  size_t n = atm2x(&ctl, &atm, NULL, NULL, NULL);
00055  size_t m = obs2y(&ctl, &obs, NULL, NULL, NULL);
00056
00057  /* Check sizes... */
00058  if (n == 0)
00059      ERRMSG("No state vector elements!");
00060  if (m == 0)
00061      ERRMSG("No measurement vector elements!");
00062
00063  /* Allocate... */
00064  k = gsl_matrix_alloc(m, n);
00065
00066  /* Compute kernel matrix... */
00067  kernel(&ctl, &atm, &obs, k);
00068
00069  /* Write matrix to file... */
00070  write_matrix(NULL, argv[4], &ctl, k, &atm, &obs, "y", "x", "r");
00071
00072  /* Free... */
00073  gsl_matrix_free(k);
00074
00075  return EXIT_SUCCESS;
00076 }

```

## 5.23 limb.c File Reference

Create observation geometry for a limb sounder.

```
#include "jurassic.h"
```

### Functions

- int [main](#) (int argc, char \*argv[])

#### 5.23.1 Detailed Description

Create observation geometry for a limb sounder.

Definition in file [limb.c](#).

#### 5.23.2 Function Documentation

**5.23.2.1 main()** `int main (`  
`int argc,`  
`char * argv[] )`

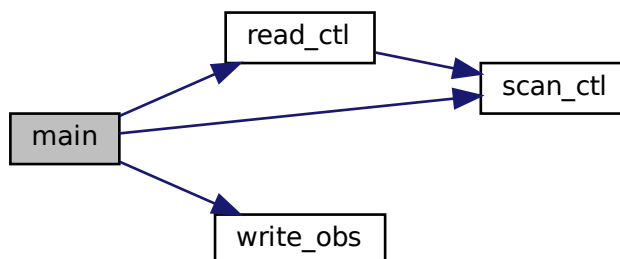
Definition at line 27 of file `limb.c`.

```

00029     {
00030
00031     static ctl_t ctl;
00032     static obs_t obs;
00033
00034     /* Check arguments... */
00035     if (argc < 3)
00036         ERRMSG("Give parameters: <ctl> <obs>");
00037
00038     /* Read control parameters... */
00039     read_ctl(argc, argv, &ctl);
00040     double obsz = scan_ctl(argc, argv, "OBSZ", -1, "780", NULL);
00041     double t0 = scan_ctl(argc, argv, "T0", -1, "0", NULL);
00042     double t1 = scan_ctl(argc, argv, "T1", -1, "0", NULL);
00043     double dt = scan_ctl(argc, argv, "DT", -1, "1", NULL);
00044     double z0 = scan_ctl(argc, argv, "Z0", -1, "3", NULL);
00045     double z1 = scan_ctl(argc, argv, "Z1", -1, "68", NULL);
00046     double dz = scan_ctl(argc, argv, "DZ", -1, "1", NULL);
00047
00048     /* Create measurement geometry... */
00049     for (double t = t0; t <= t1; t += dt)
00050         for (double z = z0; z <= z1; z += dz) {
00051             obs.time[obs.nr] = t;
00052             obs.obsz[obs.nr] = obsz;
00053             obs.vpz[obs.nr] = z;
00054             obs.vplat[obs.nr] = 180 / M_PI * acos((RE + z) / (RE + obsz));
00055             if ((++obs.nr) >= NR)
00056                 ERRMSG("Too many rays!");
00057         }
00058
00059     /* Write observation data... */
00060     write_obs(NULL, argv[2], &ctl, &obs);
00061
00062     return EXIT_SUCCESS;
00063 }

```

Here is the call graph for this function:



## 5.24 limb.c

```

00001 /*
00002  This file is part of JURASSIC.
00003
00004  JURASSIC is free software: you can redistribute it and/or modify
00005  it under the terms of the GNU General Public License as published by
00006  the Free Software Foundation, either version 3 of the License, or
00007  (at your option) any later version.
00008
00009  JURASSIC is distributed in the hope that it will be useful,
00010  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

```

```

00012  GNU General Public License for more details.
00013
00014  You should have received a copy of the GNU General Public License
00015  along with JURASSIC. If not, see <http://www.gnu.org/licenses/>.
00016
00017  Copyright (C) 2003-2021 Forschungszentrum Juelich GmbH
00018  */
00019
00025  #include "jurassic.h"
00026
00027  int main(
00028      int argc,
00029      char *argv[]) {
00030
00031      static ctl_t ctl;
00032      static obs_t obs;
00033
00034      /* Check arguments... */
00035      if (argc < 3)
00036          ERRMSG("Give parameters: <ctl> <obs>");
00037
00038      /* Read control parameters... */
00039      read_ctl(argc, argv, &ctl);
00040      double obsz = scan_ctl(argc, argv, "OBSZ", -1, "780", NULL);
00041      double t0 = scan_ctl(argc, argv, "T0", -1, "0", NULL);
00042      double t1 = scan_ctl(argc, argv, "T1", -1, "0", NULL);
00043      double dt = scan_ctl(argc, argv, "DT", -1, "1", NULL);
00044      double z0 = scan_ctl(argc, argv, "Z0", -1, "3", NULL);
00045      double z1 = scan_ctl(argc, argv, "Z1", -1, "68", NULL);
00046      double dz = scan_ctl(argc, argv, "DZ", -1, "1", NULL);
00047
00048      /* Create measurement geometry... */
00049      for (double t = t0; t <= t1; t += dt)
00050          for (double z = z0; z <= z1; z += dz) {
00051              obs.time[obs.nr] = t;
00052              obs.obsz[obs.nr] = obsz;
00053              obs.vpz[obs.nr] = z;
00054              obs.vplat[obs.nr] = 180 / M_PI * acos((RE + z) / (RE + obsz));
00055              if (++obs.nr >= NR)
00056                  ERRMSG("Too many rays!");
00057          }
00058
00059      /* Write observation data... */
00060      write_obs(NULL, argv[2], &ctl, &obs);
00061
00062      return EXIT_SUCCESS;
00063 }

```

## 5.25 nadir.c File Reference

Create observation geometry for a nadir sounder.

```
#include "jurassic.h"
```

### Functions

- int [main](#) (int argc, char \*argv[])

#### 5.25.1 Detailed Description

Create observation geometry for a nadir sounder.

Definition in file [nadir.c](#).

#### 5.25.2 Function Documentation

**5.25.2.1 main()** `int main (`  
`int argc,`  
`char * argv[] )`

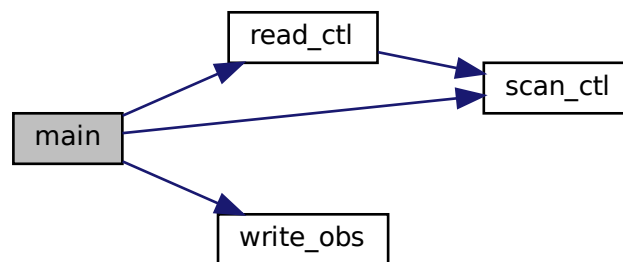
Definition at line 27 of file [nadir.c](#).

```

00029     {
00030
00031     static ctl_t ctl;
00032     static obs_t obs;
00033
00034     /* Check arguments... */
00035     if (argc < 3)
00036         ERRMSG("Give parameters: <ctl> <obs>");
00037
00038     /* Read control parameters... */
00039     read_ctl(argc, argv, &ctl);
00040     double t0 = scan_ctl(argc, argv, "T0", -1, "0", NULL);
00041     double t1 = scan_ctl(argc, argv, "T1", -1, "0", NULL);
00042     double dt = scan_ctl(argc, argv, "DT", -1, "1", NULL);
00043     double obsz = scan_ctl(argc, argv, "OBSZ", -1, "700", NULL);
00044     double lat0 = scan_ctl(argc, argv, "LAT0", -1, "-8.01", NULL);
00045     double lat1 = scan_ctl(argc, argv, "LAT1", -1, "8.01", NULL);
00046     double dlat = scan_ctl(argc, argv, "DLAT", -1, "0.18", NULL);
00047
00048     /* Create measurement geometry... */
00049     for (double t = t0; t <= t1; t += dt)
00050         for (double lat = lat0; lat <= lat1; lat += dlat) {
00051             obs.time[obs.nr] = t;
00052             obs.obsz[obs.nr] = obsz;
00053             obs.vplat[obs.nr] = lat;
00054             if (++obs.nr >= NR)
00055                 ERRMSG("Too many rays!");
00056         }
00057
00058     /* Write observation data... */
00059     write_obs(NULL, argv[2], &ctl, &obs);
00060
00061     return EXIT_SUCCESS;
00062 }

```

Here is the call graph for this function:



## 5.26 nadir.c

```

00001 /*
00002  This file is part of JURASSIC.
00003
00004  JURASSIC is free software: you can redistribute it and/or modify
00005  it under the terms of the GNU General Public License as published by
00006  the Free Software Foundation, either version 3 of the License, or
00007  (at your option) any later version.
00008
00009  JURASSIC is distributed in the hope that it will be useful,
00010  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  GNU General Public License for more details.

```

```

00013
00014 You should have received a copy of the GNU General Public License
00015 along with JURASSIC. If not, see <http://www.gnu.org/licenses/>.
00016
00017 Copyright (C) 2003-2021 Forschungszentrum Juelich GmbH
00018 */
00019
00025 #include "jurassic.h"
00026
00027 int main(
00028     int argc,
00029     char *argv[]) {
00030
00031     static ctl_t ctl;
00032     static obs_t obs;
00033
00034     /* Check arguments... */
00035     if (argc < 3)
00036         ERRMSG("Give parameters: <ctl> <obs>");
00037
00038     /* Read control parameters... */
00039     read_ctl(argc, argv, &ctl);
00040     double t0 = scan_ctl(argc, argv, "T0", -1, "0", NULL);
00041     double t1 = scan_ctl(argc, argv, "T1", -1, "0", NULL);
00042     double dt = scan_ctl(argc, argv, "DT", -1, "1", NULL);
00043     double obsz = scan_ctl(argc, argv, "OBSZ", -1, "700", NULL);
00044     double lat0 = scan_ctl(argc, argv, "LAT0", -1, "-8.01", NULL);
00045     double lat1 = scan_ctl(argc, argv, "LAT1", -1, "8.01", NULL);
00046     double dlat = scan_ctl(argc, argv, "DLAT", -1, "0.18", NULL);
00047
00048     /* Create measurement geometry... */
00049     for (double t = t0; t <= t1; t += dt)
00050         for (double lat = lat0; lat <= lat1; lat += dlat) {
00051             obs.time[obs.nr] = t;
00052             obs.obsz[obs.nr] = obsz;
00053             obs.vplat[obs.nr] = lat;
00054             if ((++obs.nr) >= NR)
00055                 ERRMSG("Too many rays!");
00056         }
00057
00058     /* Write observation data... */
00059     write_obs(NULL, argv[2], &ctl, &obs);
00060
00061     return EXIT_SUCCESS;
00062 }

```

## 5.27 obs2spec.c File Reference

Converter for spectra.

```
#include <omp.h>
#include "jurassic.h"
```

### Functions

- int [main](#) (int argc, char \*argv[])

#### 5.27.1 Detailed Description

Converter for spectra.

Definition in file [obs2spec.c](#).

#### 5.27.2 Function Documentation

```

5.27.2.1 main() int main (
                int argc,
                char * argv[] )

```

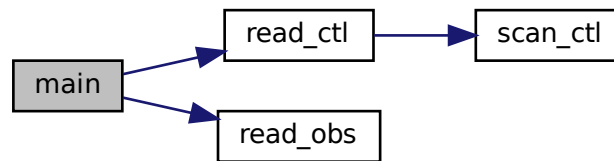
Definition at line 32 of file [obs2spec.c](#).

```

00034         {
00035
00036     static ctl_t ctl;
00037
00038     obs_t *obs;
00039
00040     FILE *out;
00041
00042     /* Check arguments... */
00043     if (argc < 4)
00044         ERRMSG("Give parameters: <ctl> <obs> <spec.tab>");
00045
00046     /* Allocate... */
00047     ALLOC(obs, obs_t, 1);
00048
00049     /* Read control parameters... */
00050     read_ctl(argc, argv, &ctl);
00051
00052     /* Read observation geometry... */
00053     read_obs(".", argv[2], &ctl, obs);
00054
00055     /* Write info... */
00056     LOG(1, "Write spectra: %s", argv[3]);
00057
00058     /* Create file... */
00059     if (!(out = fopen(argv[3], "w")))
00060         ERRMSG("Cannot create file!");
00061
00062     /* Write header... */
00063     fprintf(out,
00064             "# $1 = time (seconds since 2000-01-01T00:00Z)\n"
00065             "# $2 = observer altitude [km]\n"
00066             "# $3 = observer longitude [deg]\n"
00067             "# $4 = observer latitude [deg]\n"
00068             "# $5 = view point altitude [km]\n"
00069             "# $6 = view point longitude [deg]\n"
00070             "# $7 = view point latitude [deg]\n"
00071             "# $8 = tangent point altitude [km]\n"
00072             "# $9 = tangent point longitude [deg]\n"
00073             "# $10 = tangent point latitude [deg]\n"
00074             "# $11 = channel frequency [cm^-1]\n"
00075             "# $12 = channel radiance [W/(m^2 sr cm^-1)]\n"
00076             "# $13 = channel transmittance [1]\n");
00077
00078     /* Write data... */
00079     for (int ir = 0; ir < obs->nr; ir++) {
00080         fprintf(out, "\n");
00081         for (int id = 0; id < ctl.nd; id++)
00082             fprintf(out, "%.2f %g %g %g %g %g %g %g %g %.4f %g %g\n",
00083                     obs->time[ir], obs->obsz[ir], obs->obslon[ir], obs->obslat[ir],
00084                     obs->vpz[ir], obs->vp lon[ir], obs->vplat[ir], obs->tpz[ir],
00085                     obs->tp lon[ir], obs->tplat[ir], ctl.nu[id], obs->rad[id][ir],
00086                     obs->tau[id][ir]);
00087     }
00088
00089     /* Close file... */
00090     fclose(out);
00091
00092     /* Free... */
00093     free(obs);
00094
00095     return EXIT_SUCCESS;
00096 }

```

Here is the call graph for this function:



## 5.28 obs2spec.c

```

00001 /*
00002  This file is part of JURASSIC.
00003
00004  JURASSIC is free software: you can redistribute it and/or modify
00005  it under the terms of the GNU General Public License as published by
00006  the Free Software Foundation, either version 3 of the License, or
00007  (at your option) any later version.
00008
00009  JURASSIC is distributed in the hope that it will be useful,
00010  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  GNU General Public License for more details.
00013
00014  You should have received a copy of the GNU General Public License
00015  along with JURASSIC. If not, see <http://www.gnu.org/licenses/>.
00016
00017  Copyright (C) 2003-2021 Forschungszentrum Juelich GmbH
00018 */
00019
00025 #include <omp.h>
00026 #include "jurassic.h"
00027
00028 /* -----
00029  Main...
00030  ----- */
00031
00032 int main(
00033     int argc,
00034     char *argv[]) {
00035
00036     static ctl_t ctl;
00037
00038     obs_t *obs;
00039
00040     FILE *out;
00041
00042     /* Check arguments... */
00043     if (argc < 4)
00044         ERRMSG("Give parameters: <ctl> <obs> <spec.tab>");
00045
00046     /* Allocate... */
00047     ALLOC(obs, obs_t, 1);
00048
00049     /* Read control parameters... */
00050     read_ctl(argc, argv, &ctl);
00051
00052     /* Read observation geometry... */
00053     read_obs(".", argv[2], &ctl, obs);
00054
00055     /* Write info... */
00056     LOG(1, "Write spectra: %s", argv[3]);
00057
00058     /* Create file... */
00059     if (!(out = fopen(argv[3], "w")))
00060         ERRMSG("Cannot create file!");
00061
00062     /* Write header... */
00063     fprintf(out,
00064         "# $1 = time (seconds since 2000-01-01T00:00Z)\n"
00065         "# $2 = observer altitude [km]\n"

```



```

00066         "# $3 = observer longitude [deg]\n"
00067         "# $4 = observer latitude [deg]\n"
00068         "# $5 = view point altitude [km]\n"
00069         "# $6 = view point longitude [deg]\n"
00070         "# $7 = view point latitude [deg]\n"
00071         "# $8 = tangent point altitude [km]\n"
00072         "# $9 = tangent point longitude [deg]\n"
00073         "# $10 = tangent point latitude [deg]\n"
00074         "# $11 = channel frequency [cm^-1]\n"
00075         "# $12 = channel radiance [W/(m^2 sr cm^-1)]\n"
00076         "# $13 = channel transmittance [1]\n");
00077
00078     /* Write data... */
00079     for (int ir = 0; ir < obs->nr; ir++) {
00080         fprintf(out, "\n");
00081         for (int id = 0; id < ctl.nd; id++)
00082             fprintf(out, "%.2f %g %g %g %g %g %g %g %g %.4f %g %g\n",
00083                 obs->time[ir], obs->obsz[ir], obs->obslon[ir], obs->obslat[ir],
00084                 obs->vpz[ir], obs->vplon[ir], obs->vplat[ir], obs->tpz[ir],
00085                 obs->tplon[ir], obs->tplat[ir], ctl.nu[id], obs->rad[id][ir],
00086                 obs->tau[id][ir]);
00087     }
00088
00089     /* Close file... */
00090     fclose(out);
00091
00092     /* Free... */
00093     free(obs);
00094
00095     return EXIT_SUCCESS;
00096 }

```

## 5.29 planck.c File Reference

Convert brightness temperature to radiance.

```
#include "jurassic.h"
```

### Functions

- int [main](#) (int argc, char \*argv[])

#### 5.29.1 Detailed Description

Convert brightness temperature to radiance.

Definition in file [planck.c](#).

#### 5.29.2 Function Documentation

```

5.29.2.1 main() int main (
                int argc,
                char * argv[] )

```

Definition at line 27 of file [planck.c](#).

```

00029     {
00030
00031     /* Check arguments... */
00032     if (argc != 3 && argc != 7)
00033         ERRMSG
00034         ("Give parameters: [ <t> <nu> | <t0> <t1> <dt> <nu0> <nu1> <dnu> ]");
00035
00036     /* Calculate single value... */
00037     if (argc == 3) {
00038
00039         /* Read arguments... */
00040         double t = atof(argv[1]);
00041         double nu = atof(argv[2]);
00042
00043         /* Compute Planck function... */
00044         printf("%.10g\n", planck(t, nu));
00045     }
00046
00047     /* Calculate table... */
00048     else if (argc == 7) {
00049
00050         /* Read arguments... */
00051         double t0 = atof(argv[1]);
00052         double t1 = atof(argv[2]);
00053         double dt = atof(argv[3]);
00054         double nu0 = atof(argv[4]);
00055         double nu1 = atof(argv[5]);
00056         double dnu = atof(argv[6]);
00057
00058         /* Write header... */
00059         printf("# $1 = brightness temperature [K]\n"
00060                "# $2 = wavenumber [cm^-1]\n"
00061                "# $3 = radiance [W/(m^2 sr cm^-1)]\n");
00062
00063         /* Compute Planck function... */
00064         for (double t = t0; t <= t1; t += dt) {
00065             printf("\n");
00066             for (double nu = nu0; nu <= nu1; nu += dnu)
00067                 printf("%.10g %.4f %.10g\n", t, nu, planck(t, nu));
00068         }
00069     }
00070
00071     return EXIT_SUCCESS;
00072 }

```

Here is the call graph for this function:



## 5.30 planck.c

```

00001 /*
00002  This file is part of JURASSIC.
00003
00004  JURASSIC is free software: you can redistribute it and/or modify
00005  it under the terms of the GNU General Public License as published by
00006  the Free Software Foundation, either version 3 of the License, or
00007  (at your option) any later version.
00008
00009  JURASSIC is distributed in the hope that it will be useful,
00010  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  GNU General Public License for more details.

```

```

00013
00014 You should have received a copy of the GNU General Public License
00015 along with JURASSIC. If not, see <http://www.gnu.org/licenses/>.
00016
00017 Copyright (C) 2003-2021 Forschungszentrum Juelich GmbH
00018 */
00019
00025 #include "jurassic.h"
00026
00027 int main(
00028     int argc,
00029     char *argv[]) {
00030
00031     /* Check arguments... */
00032     if (argc != 3 && argc != 7)
00033         ERRMSG
00034         ("Give parameters: [ <t> <nu> | <t0> <t1> <dt> <nu0> <nu1> <dnu> ]");
00035
00036     /* Calculate single value... */
00037     if (argc == 3) {
00038
00039         /* Read arguments... */
00040         double t = atof(argv[1]);
00041         double nu = atof(argv[2]);
00042
00043         /* Compute Planck function... */
00044         printf("%.10g\n", planck(t, nu));
00045     }
00046
00047     /* Calculate table... */
00048     else if (argc == 7) {
00049
00050         /* Read arguments... */
00051         double t0 = atof(argv[1]);
00052         double t1 = atof(argv[2]);
00053         double dt = atof(argv[3]);
00054         double nu0 = atof(argv[4]);
00055         double nu1 = atof(argv[5]);
00056         double dnu = atof(argv[6]);
00057
00058         /* Write header... */
00059         printf("# $1 = brightness temperature [K]\n"
00060             "# $2 = wavenumber [cm^-1]\n"
00061             "# $3 = radiance [W/(m^2 sr cm^-1)]\n");
00062
00063         /* Compute Planck function... */
00064         for (double t = t0; t <= t1; t += dt) {
00065             printf("\n");
00066             for (double nu = nu0; nu <= nu1; nu += dnu)
00067                 printf("%.10g %.4f %.10g\n", t, nu, planck(t, nu));
00068         }
00069     }
00070
00071     return EXIT_SUCCESS;
00072 }

```

## 5.31 raytrace.c File Reference

Determine atmospheric ray paths.

```
#include "jurassic.h"
```

### Functions

- int [main](#) (int argc, char \*argv[])

#### 5.31.1 Detailed Description

Determine atmospheric ray paths.

Definition in file [raytrace.c](#).

## 5.31.2 Function Documentation

**5.31.2.1 main()** `int main (`  
`int argc,`  
`char * argv[] )`

Definition at line 27 of file [raytrace.c](#).

```

00029     {
00030
00031     static atm_t atm, atm2;
00032     static ctl_t ctl;
00033     static los_t los;
00034     static obs_t obs;
00035
00036     FILE *out;
00037
00038     char filename[LEN], losbase[LEN];
00039
00040     double u[NG];
00041
00042     /* Check arguments... */
00043     if (argc < 4)
00044         ERRMSG("Give parameters: <ctl> <obs> <atm>");
00045
00046     /* Read control parameters... */
00047     read_ctl(argc, argv, &ctl);
00048
00049     /* Get basenames... */
00050     scan_ctl(argc, argv, "LOSBASE", -1, "los", losbase);
00051
00052     /* Read observation geometry... */
00053     read_obs(NULL, argv[2], &ctl, &obs);
00054
00055     /* Read atmospheric data... */
00056     read_atm(NULL, argv[3], &ctl, &atm);
00057
00058     /* Write info... */
00059     LOG(1, "Write raytrace data: raytrace.tab");
00060
00061     /* Create file... */
00062     if (!(out = fopen("raytrace.tab", "w")))
00063         ERRMSG("Cannot create file!");
00064
00065     /* Write header... */
00066     fprintf(out,
00067         "# $1 = time (seconds since 2000-01-01T00:00Z)\n"
00068         "# $2 = observer altitude [km]\n"
00069         "# $3 = observer longitude [deg]\n"
00070         "# $4 = observer latitude [deg]\n"
00071         "# $5 = view point altitude [km]\n"
00072         "# $6 = view point longitude [deg]\n"
00073         "# $7 = view point latitude [deg]\n"
00074         "# $8 = tangent point altitude [km]\n"
00075         "# $9 = tangent point longitude [deg]\n"
00076         "# $10 = tangent point latitude [deg]\n"
00077         "# $11 = ray path index\n" "# $12 = ray path length [km]\n");
00078     for (int ig = 0; ig < ctl.ng; ig++)
00079         fprintf(out, "# $%d = %s column density [molec/cm^2]\n",
00080             13 + ig, ctl.emitter[ig]);
00081     fprintf(out, "\n");
00082
00083     /* Loop over rays... */
00084     for (int ir = 0; ir < obs.nr; ir++) {
00085
00086         /* Raytracing... */
00087         raytrace(&ctl, &atm, &obs, &los, ir);
00088
00089         /* Copy data... */
00090         atm2.np = los.np;
00091         for (int ip = 0; ip < los.np; ip++) {
00092             atm2.time[ip] = obs.time[ir];
00093             atm2.z[ip] = los.z[ip];
00094             atm2.lon[ip] = los.lon[ip];
00095             atm2.lat[ip] = los.lat[ip];
00096             atm2.p[ip] = los.p[ip];
00097             atm2.t[ip] = los.t[ip];
00098             for (int ig = 0; ig < ctl.ng; ig++)
00099                 atm2.q[ig][ip] = los.q[ip][ig];
00100             for (int iw = 0; iw < ctl.nw; iw++)

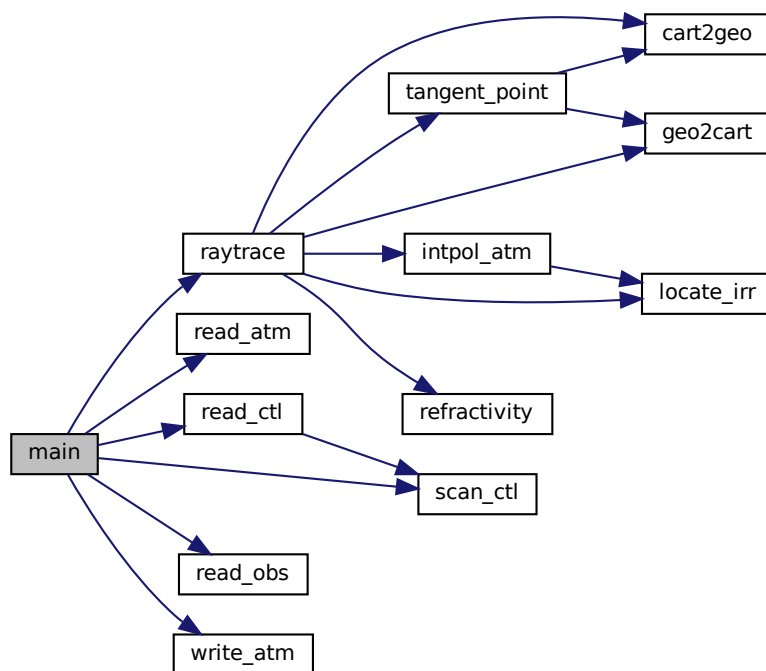
```

```

00101     atm2.k[iw][ip] = GSL_NAN;
00102 }
00103
00104 /* Save data... */
00105 sprintf(filename, "los.%d", ir);
00106 write_atm(NULL, filename, &ctl, &atm2);
00107
00108 /* Get column densities... */
00109 double s = 0;
00110 for (int ig = 0; ig < ctl.ng; ig++)
00111     u[ig] = 0;
00112 for (int ip = 0; ip < los.np; ip++) {
00113     s += los.ds[ip];
00114     for (int ig = 0; ig < ctl.ng; ig++)
00115         u[ig] += los.u[ip][ig];
00116 }
00117
00118 /* Write summary data... */
00119 fprintf(out, "%.2f %g %g %g %g %g %g %g %g %g %g",
00120         obs.time[ir], obs.obsz[ir], obs.obslon[ir], obs.obslat[ir],
00121         obs.vpz[ir], obs.vplon[ir], obs.vplat[ir],
00122         obs.tpz[ir], obs.tplon[ir], obs.tplat[ir], ir, s);
00123 for (int ig = 0; ig < ctl.ng; ig++)
00124     fprintf(out, " %g", u[ig]);
00125 fprintf(out, "\n");
00126 }
00127
00128 /* Close file... */
00129 fclose(out);
00130
00131 return EXIT_SUCCESS;
00132 }

```

Here is the call graph for this function:



## 5.32 raytrace.c

```

00001 /*
00002  This file is part of JURASSIC.
00003
00004  JURASSIC is free software: you can redistribute it and/or modify

```

```

00005  it under the terms of the GNU General Public License as published by
00006  the Free Software Foundation, either version 3 of the License, or
00007  (at your option) any later version.
00008
00009  JURASSIC is distributed in the hope that it will be useful,
00010  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  GNU General Public License for more details.
00013
00014  You should have received a copy of the GNU General Public License
00015  along with JURASSIC. If not, see <http://www.gnu.org/licenses/>.
00016
00017  Copyright (C) 2003-2021 Forschungszentrum Juelich GmbH
00018  */
00019
00025  #include "jurassic.h"
00026
00027  int main(
00028      int argc,
00029      char *argv[]) {
00030
00031      static atm_t atm, atm2;
00032      static ctl_t ctl;
00033      static los_t los;
00034      static obs_t obs;
00035
00036      FILE *out;
00037
00038      char filename[LEN], losbase[LEN];
00039
00040      double u[NG];
00041
00042      /* Check arguments... */
00043      if (argc < 4)
00044          ERRMSG("Give parameters: <ctl> <obs> <atm>");
00045
00046      /* Read control parameters... */
00047      read_ctl(argc, argv, &ctl);
00048
00049      /* Get basenames... */
00050      scan_ctl(argc, argv, "LOSBASE", -1, "los", losbase);
00051
00052      /* Read observation geometry... */
00053      read_obs(NULL, argv[2], &ctl, &obs);
00054
00055      /* Read atmospheric data... */
00056      read_atm(NULL, argv[3], &ctl, &atm);
00057
00058      /* Write info... */
00059      LOG(1, "Write raytrace data: raytrace.tab");
00060
00061      /* Create file... */
00062      if (!(out = fopen("raytrace.tab", "w")))
00063          ERRMSG("Cannot create file!");
00064
00065      /* Write header... */
00066      fprintf(out,
00067          "# $1 = time (seconds since 2000-01-01T00:00Z)\n"
00068          "# $2 = observer altitude [km]\n"
00069          "# $3 = observer longitude [deg]\n"
00070          "# $4 = observer latitude [deg]\n"
00071          "# $5 = view point altitude [km]\n"
00072          "# $6 = view point longitude [deg]\n"
00073          "# $7 = view point latitude [deg]\n"
00074          "# $8 = tangent point altitude [km]\n"
00075          "# $9 = tangent point longitude [deg]\n"
00076          "# $10 = tangent point latitude [deg]\n"
00077          "# $11 = ray path index\n" "# $12 = ray path length [km]\n");
00078      for (int ig = 0; ig < ctl.ng; ig++)
00079          fprintf(out, "# $11 = %s column density [molec/cm^2]\n",
00080              13 + ig, ctl.emitter[ig]);
00081      fprintf(out, "\n");
00082
00083      /* Loop over rays... */
00084      for (int ir = 0; ir < obs.nr; ir++) {
00085
00086          /* Raytracing... */
00087          raytrace(&ctl, &atm, &obs, &los, ir);
00088
00089          /* Copy data... */
00090          atm2.np = los.np;
00091          for (int ip = 0; ip < los.np; ip++) {
00092              atm2.time[ip] = obs.time[ir];
00093              atm2.z[ip] = los.z[ip];
00094              atm2.lon[ip] = los.lon[ip];
00095              atm2.lat[ip] = los.lat[ip];
00096              atm2.p[ip] = los.p[ip];

```

```

00097     atm2.t[ip] = los.t[ip];
00098     for (int ig = 0; ig < ctl.ng; ig++)
00099         atm2.q[ig][ip] = los.q[ip][ig];
00100     for (int iw = 0; iw < ctl.nw; iw++)
00101         atm2.k[iw][ip] = GSL_NAN;
00102 }
00103
00104 /* Save data... */
00105 sprintf(filename, "los.%d", ir);
00106 write_atm(NULL, filename, &ctl, &atm2);
00107
00108 /* Get column densities... */
00109 double s = 0;
00110 for (int ig = 0; ig < ctl.ng; ig++)
00111     u[ig] = 0;
00112 for (int ip = 0; ip < los.np; ip++) {
00113     s += los.ds[ip];
00114     for (int ig = 0; ig < ctl.ng; ig++)
00115         u[ig] += los.u[ip][ig];
00116 }
00117
00118 /* Write summary data... */
00119 fprintf(out, "%.2f %g %g %g %g %g %g %g %g %d %g",
00120         obs.time[ir], obs.obsz[ir], obs.obslon[ir], obs.obslat[ir],
00121         obs.vpz[ir], obs.vplon[ir], obs.vplat[ir],
00122         obs.tpz[ir], obs.tplon[ir], obs.tplat[ir], ir, s);
00123 for (int ig = 0; ig < ctl.ng; ig++)
00124     fprintf(out, " %g", u[ig]);
00125 fprintf(out, "\n");
00126 }
00127
00128 /* Close file... */
00129 fclose(out);
00130
00131 return EXIT_SUCCESS;
00132 }

```

### 5.33 retrieval.c File Reference

JURASSIC retrieval processor.

```
#include "jurassic.h"
```

#### Data Structures

- struct [ret\\_t](#)  
*Retrieval control parameters.*

#### Functions

- void [analyze\\_avk](#) ([ret\\_t](#) \*ret, [ctl\\_t](#) \*ctl, [atm\\_t](#) \*atm, int \*iqa, int \*ipa, gsl\_matrix \*avk)  
*Compute information content and resolution.*
- void [analyze\\_avk\\_quantity](#) (gsl\_matrix \*avk, int iq, int ipa, size\_t \*n0, size\_t \*n1, double \*cont, double \*res)  
*Analyze averaging kernels for individual retrieval target.*
- double [cost\\_function](#) (gsl\_vector \*dx, gsl\_vector \*dy, gsl\_matrix \*s\_a\_inv, gsl\_vector \*sig\_eps\_inv)  
*Compute cost function.*
- void [matrix\\_invert](#) (gsl\_matrix \*a)  
*Invert symmetric matrix.*
- void [matrix\\_product](#) (gsl\_matrix \*a, gsl\_vector \*b, int transpose, gsl\_matrix \*c)  
*Compute matrix product  $A^T B$  or  $ABA^T$  for diagonal matrix  $B$ .*
- void [optimal\\_estimation](#) ([ret\\_t](#) \*ret, [ctl\\_t](#) \*ctl, [obs\\_t](#) \*obs\_meas, [obs\\_t](#) \*obs\_i, [atm\\_t](#) \*atm\_apr, [atm\\_t](#) \*atm\_i)  
*Carry out optimal estimation retrieval.*
- void [read\\_ret](#) (int argc, char \*argv[], [ctl\\_t](#) \*ctl, [ret\\_t](#) \*ret)

*Read retrieval control parameters.*

- void [set\\_cov\\_apr](#) ([ret\\_t](#) \*ret, [ctl\\_t](#) \*ctl, [atm\\_t](#) \*atm, int \*iqa, int \*ipa, [gsl\\_matrix](#) \*s\_a)

*Set a priori covariance.*

- void [set\\_cov\\_meas](#) ([ret\\_t](#) \*ret, [ctl\\_t](#) \*ctl, [obs\\_t](#) \*obs, [gsl\\_vector](#) \*sig\_noise, [gsl\\_vector](#) \*sig\_formod, [gsl\\_vector](#) \*sig\_eps\_inv)

*Set measurement errors.*

- void [write\\_stddev](#) (const char \*quantity, [ret\\_t](#) \*ret, [ctl\\_t](#) \*ctl, [atm\\_t](#) \*atm, [gsl\\_matrix](#) \*s)

*Write retrieval error to file.*

- int [main](#) (int argc, char \*argv[])

### 5.33.1 Detailed Description

JURASSIC retrieval processor.

Definition in file [retrieval.c](#).

### 5.33.2 Function Documentation

**5.33.2.1 [analyze\\_avk\(\)](#)** void [analyze\\_avk](#) (  
[ret\\_t](#) \* ret,  
[ctl\\_t](#) \* ctl,  
[atm\\_t](#) \* atm,  
int \* iqa,  
int \* ipa,  
[gsl\\_matrix](#) \* avk )

Compute information content and resolution.

Definition at line 257 of file [retrieval.c](#).

```
00263     {
00264
00265     static atm\_t atm_cont, atm_res;
00266
00267     int icl, ig, iq, isf, iw;
00268
00269     size_t i, n, n0[NQ], n1[NQ];
00270
00271     /* Get sizes... */
00272     n = avk->size1;
00273
00274     /* Find sub-matrices for different quantities... */
00275     for (iq = 0; iq < NQ; iq++) {
00276         n0[iq] = N;
00277         for (i = 0; i < n; i++) {
00278             if (iqa[i] == iq && n0[iq] == N)
00279                 n0[iq] = i;
00280             if (iqa[i] == iq)
00281                 n1[iq] = i - n0[iq] + 1;
00282         }
00283     }
00284
00285     /* Initialize... */
00286     copy\_atm(ctl, &atm_cont, atm, 1);
00287     copy\_atm(ctl, &atm_res, atm, 1);
00288
00289     /* Analyze quantities... */
00290     analyze\_avk\_quantity(avk, IDXP, ipa, n0, n1, atm_cont.p, atm_res.p);
00291     analyze\_avk\_quantity(avk, IDXT, ipa, n0, n1, atm_cont.t, atm_res.t);
00292     for (ig = 0; ig < ctl->ng; ig++)
00293         analyze\_avk\_quantity(avk, IDXQ(ig), ipa, n0, n1,
00294                             atm_cont.q[ig], atm_res.q[ig]);
00295     for (iw = 0; iw < ctl->nw; iw++)
```

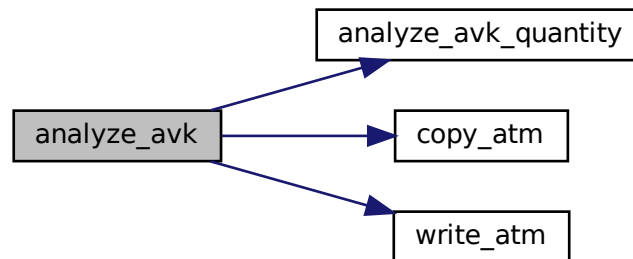


```

00296     analyze_avk_quantity(avk, IDXK(iw), ipa, n0, n1,
00297                          atm_cont.k[iw], atm_res.k[iw]);
00298     analyze_avk_quantity(avk, IDXCLZ, ipa, n0, n1, &atm_cont.clz, &atm_res.clz);
00299     analyze_avk_quantity(avk, IDXCLDZ, ipa, n0, n1, &atm_cont.cldz,
00300                          &atm_res.cldz);
00301     for (icl = 0; icl < ctl->ncl; icl++)
00302         analyze_avk_quantity(avk, IDXCLK(icl), ipa, n0, n1,
00303                              &atm_cont.clk[icl], &atm_res.clk[icl]);
00304     analyze_avk_quantity(avk, IDXSFZ, ipa, n0, n1, &atm_cont.sfz, &atm_res.sfz);
00305     analyze_avk_quantity(avk, IDXSFP, ipa, n0, n1, &atm_cont.sfp, &atm_res.sfp);
00306     analyze_avk_quantity(avk, IDXSFT, ipa, n0, n1, &atm_cont.sft, &atm_res.sft);
00307     for (isf = 0; isf < ctl->nsf; isf++)
00308         analyze_avk_quantity(avk, IDXSFEPS(isf), ipa, n0, n1,
00309                              &atm_cont.sfeps[isf], &atm_res.sfeps[isf]);
00310
00311     /* Write results to disk... */
00312     write_atm(ret->dir, "atm_cont.tab", ctl, &atm_cont);
00313     write_atm(ret->dir, "atm_res.tab", ctl, &atm_res);
00314 }

```

Here is the call graph for this function:



**5.33.2.2 analyze\_avk\_quantity()** void analyze\_avk\_quantity (

```

    gsl_matrix * avk,
    int iq,
    int * ipa,
    size_t * n0,
    size_t * n1,
    double * cont,
    double * res )

```

Analyze averaging kernels for individual retrieval target.

Definition at line 318 of file [retrieval.c](#).

```

00325     {
00326
00327     /* Loop over state vector elements... */
00328     if (n0[iq] < N)
00329         for (size_t i = 0; i < n1[iq]; i++) {
00330
00331         /* Get area of averaging kernel... */
00332         for (size_t j = 0; j < n1[iq]; j++)
00333             cont[ipa[n0[iq] + i]] += gsl_matrix_get(avk, n0[iq] + i, n0[iq] + j);
00334
00335         /* Get information density... */
00336         res[ipa[n0[iq] + i]] = 1 / gsl_matrix_get(avk, n0[iq] + i, n0[iq] + i);
00337     }
00338 }

```

**5.33.2.3 cost\_function()** double cost\_function (

```

    gsl_vector * dx,
    gsl_vector * dy,
    gsl_matrix * s_a_inv,
    gsl_vector * sig_eps_inv )

```

Compute cost function.

Definition at line 342 of file [retrieval.c](#).

```

00346         {
00347
00348     gsl_vector *x_aux, *y_aux;
00349
00350     double chisq_a, chisq_m = 0;
00351
00352     /* Get sizes... */
00353     size_t m = dy->size;
00354     size_t n = dx->size;
00355
00356     /* Allocate... */
00357     x_aux = gsl_vector_alloc(n);
00358     y_aux = gsl_vector_alloc(m);
00359
00360     /* Determine normalized cost function...
00361     (chi^2 = 1/m * [dy^T * S_eps^{-1} * dy + dx^T * S_a^{-1} * dx]) */
00362     for (size_t i = 0; i < m; i++)
00363         chisq_m += POW2(gsl_vector_get(dy, i) * gsl_vector_get(sig_eps_inv, i));
00364     gsl_blas_dgemv(CblasNoTrans, 1.0, s_a_inv, dx, 0.0, x_aux);
00365     gsl_blas_ddot(dx, x_aux, &chisq_a);
00366
00367     /* Free... */
00368     gsl_vector_free(x_aux);
00369     gsl_vector_free(y_aux);
00370
00371     /* Return cost function value... */
00372     return (chisq_m + chisq_a) / (double) m;
00373 }

```

**5.33.2.4 matrix\_invert()** void matrix\_invert (

```

    gsl_matrix * a )

```

Invert symmetric matrix.

Definition at line 377 of file [retrieval.c](#).

```

00378         {
00379
00380     size_t diag = 1;
00381
00382     /* Get size... */
00383     size_t n = a->size1;
00384
00385     /* Check if matrix is diagonal... */
00386     for (size_t i = 0; i < n && diag; i++)
00387         for (size_t j = i + 1; j < n; j++)
00388             if (gsl_matrix_get(a, i, j) != 0) {
00389                 diag = 0;
00390                 break;
00391             }
00392
00393     /* Quick inversion of diagonal matrix... */
00394     if (diag)
00395         for (size_t i = 0; i < n; i++)
00396             gsl_matrix_set(a, i, i, 1 / gsl_matrix_get(a, i, i));
00397
00398     /* Matrix inversion by means of Cholesky decomposition... */
00399     else {
00400         gsl_linalg_cholesky_decomp(a);
00401         gsl_linalg_cholesky_invert(a);
00402     }
00403 }

```

**5.33.2.5 matrix\_product()** void matrix\_product (

```

    gsl_matrix * a,
    gsl_vector * b,
    int transpose,
    gsl_matrix * c )

```

Compute matrix product  $A^T B A$  or  $A B A^T$  for diagonal matrix B.

Definition at line 407 of file [retrieval.c](#).

```

00411     {
00412
00413     gsl_matrix *aux;
00414
00415     /* Set sizes... */
00416     size_t m = a->size1;
00417     size_t n = a->size2;
00418
00419     /* Allocate... */
00420     aux = gsl_matrix_alloc(m, n);
00421
00422     /* Compute A^T B A... */
00423     if (transpose == 1) {
00424
00425         /* Compute B^1/2 A... */
00426         for (size_t i = 0; i < m; i++)
00427             for (size_t j = 0; j < n; j++)
00428                 gsl_matrix_set(aux, i, j,
00429                               gsl_vector_get(b, i) * gsl_matrix_get(a, i, j));
00430
00431         /* Compute A^T B A = (B^1/2 A)^T (B^1/2 A)... */
00432         gsl_blas_dgemm(CblasTrans, CblasNoTrans, 1.0, aux, aux, 0.0, c);
00433     }
00434
00435     /* Compute A B A^T... */
00436     else if (transpose == 2) {
00437
00438         /* Compute A B^1/2... */
00439         for (size_t i = 0; i < m; i++)
00440             for (size_t j = 0; j < n; j++)
00441                 gsl_matrix_set(aux, i, j,
00442                               gsl_matrix_get(a, i, j) * gsl_vector_get(b, j));
00443
00444         /* Compute A B A^T = (A B^1/2) (A B^1/2)^T... */
00445         gsl_blas_dgemm(CblasNoTrans, CblasTrans, 1.0, aux, aux, 0.0, c);
00446     }
00447
00448     /* Free... */
00449     gsl_matrix_free(aux);
00450 }

```

**5.33.2.6 optimal\_estimation()** void optimal\_estimation (

```

    ret_t * ret,
    ctl_t * ctl,
    obs_t * obs_meas,
    obs_t * obs_i,
    atm_t * atm_apr,
    atm_t * atm_i )

```

Carry out optimal estimation retrieval.

Definition at line 454 of file [retrieval.c](#).

```

00460     {
00461
00462     static int ipa[N], iqa[N];
00463
00464     gsl_matrix *a, *auxnm, *corr, *cov, *gain, *k_i, *s_a_inv;
00465     gsl_vector *b, *dx, *dy, *sig_eps_inv, *sig_formod, *sig_noise,
00466               *x_a, *x_i, *x_step, *y_aux, *y_i, *y_m;
00467
00468     FILE *out;
00469
00470     char filename[2 * LEN];
00471

```

```

00472 double chisq, chisq_old, disq = 0, lmpar = 0.001;
00473
00474 int icl, ig, ip, isf, it = 0, it2, iw;
00475
00476 size_t i, j, m, n;
00477
00478 /* -----
00479    Initialize...
00480    ----- */
00481
00482 /* Get sizes... */
00483 m = obs2y(ctl, obs_meas, NULL, NULL, NULL);
00484 n = atm2x(ctl, atm_apr, NULL, iqa, ipa);
00485 if (m == 0 || n == 0)
00486     ERRMSG("Check problem definition!");
00487
00488 /* Write info... */
00489 LOG(1, "Problem size: m= %d / n= %d "
00490      " (alloc= %.4g MB / stat= %.4g MB)",
00491      (int) m, (int) n,
00492      (double) (3 * m * n + 4 * n * n + 8 * m +
00493               8 * n) * sizeof(double) / 1024. / 1024.,
00494      (double) (5 * sizeof(atm_t) + 3 * sizeof(obs_t)
00495               + 2 * N * sizeof(int)) / 1024. / 1024.);
00496
00497 /* Allocate... */
00498 a = gsl_matrix_alloc(n, n);
00499 cov = gsl_matrix_alloc(n, n);
00500 k_i = gsl_matrix_alloc(m, n);
00501 s_a_inv = gsl_matrix_alloc(n, n);
00502
00503 b = gsl_vector_alloc(n);
00504 dx = gsl_vector_alloc(n);
00505 dy = gsl_vector_alloc(m);
00506 sig_eps_inv = gsl_vector_alloc(m);
00507 sig_formod = gsl_vector_alloc(m);
00508 sig_noise = gsl_vector_alloc(m);
00509 x_a = gsl_vector_alloc(n);
00510 x_i = gsl_vector_alloc(n);
00511 x_step = gsl_vector_alloc(n);
00512 y_aux = gsl_vector_alloc(m);
00513 y_i = gsl_vector_alloc(m);
00514 y_m = gsl_vector_alloc(m);
00515
00516 /* Set initial state... */
00517 copy_atm(ctl, atm_i, atm_apr, 0);
00518 copy_obs(ctl, obs_i, obs_meas, 0);
00519 formod(ctl, atm_i, obs_i);
00520
00521 /* Set state vectors and observation vectors... */
00522 atm2x(ctl, atm_apr, x_a, NULL, NULL);
00523 atm2x(ctl, atm_i, x_i, NULL, NULL);
00524 obs2y(ctl, obs_meas, y_m, NULL, NULL);
00525 obs2y(ctl, obs_i, y_i, NULL, NULL);
00526
00527 /* Set inverse a priori covariance S_a^-1... */
00528 set_cov_apr(ret, ctl, atm_apr, iqa, ipa, s_a_inv);
00529 write_matrix(ret->dir, "matrix_cov_apr.tab", ctl, s_a_inv,
00530             atm_i, obs_i, "x", "x", "r");
00531 matrix_invert(s_a_inv);
00532
00533 /* Get measurement errors... */
00534 set_cov_meas(ret, ctl, obs_meas, sig_noise, sig_formod, sig_eps_inv);
00535
00536 /* Create cost function file... */
00537 sprintf(filename, "%s/costs.tab", ret->dir);
00538 if (!(out = fopen(filename, "w")))
00539     ERRMSG("Cannot create cost function file!");
00540
00541 /* Write header... */
00542 fprintf(out,
00543         "# $1 = iteration number\n"
00544         "# $2 = normalized cost function\n"
00545         "# $3 = number of measurements\n"
00546         "# $4 = number of state vector elements\n\n");
00547
00548 /* Determine dx = x_i - x_a and dy = y - F(x_i) ... */
00549 gsl_vector_memcpy(dx, x_i);
00550 gsl_vector_sub(dx, x_a);
00551 gsl_vector_memcpy(dy, y_m);
00552 gsl_vector_sub(dy, y_i);
00553
00554 /* Compute cost function... */
00555 chisq = cost_function(dx, dy, s_a_inv, sig_eps_inv);
00556
00557 /* Write info... */
00558 LOG(1, "it= %d / chi^2/m= %g", it, chisq);

```

```

00559
00560 /* Write to cost function file... */
00561 fprintf(out, "%d %g %d %d\n", it, chisq, (int) m, (int) n);
00562
00563 /* Compute initial kernel... */
00564 kernel(ctl, atm_i, obs_i, k_i);
00565
00566 /* -----
00567      Levenberg-Marquardt minimization...
00568      ----- */
00569
00570 /* Outer loop... */
00571 for (it = 1; it <= ret->conv_itmax; it++) {
00572
00573     /* Store current cost function value... */
00574     chisq_old = chisq;
00575
00576     /* Compute kernel matrix K_i... */
00577     if (it > 1 && it % ret->kernel_recomp == 0)
00578         kernel(ctl, atm_i, obs_i, k_i);
00579
00580     /* Compute K_i^T * S_eps^{-1} * K_i ... */
00581     if (it == 1 || it % ret->kernel_recomp == 0)
00582         matrix_product(k_i, sig_eps_inv, 1, cov);
00583
00584     /* Determine b = K_i^T * S_eps^{-1} * dy - S_a^{-1} * dx ... */
00585     for (i = 0; i < m; i++)
00586         gsl_vector_set(y_aux, i, gsl_vector_get(dy, i)
00587             * POW2(gsl_vector_get(sig_eps_inv, i)));
00588     gsl_blas_dgemv(CblasTrans, 1.0, k_i, y_aux, 0.0, b);
00589     gsl_blas_dgemv(CblasNoTrans, -1.0, s_a_inv, dx, 1.0, b);
00590
00591     /* Inner loop... */
00592     for (it2 = 0; it2 < 20; it2++) {
00593
00594         /* Compute A = (1 + lmpar) * S_a^{-1} + K_i^T * S_eps^{-1} * K_i ... */
00595         gsl_matrix_memcpy(a, s_a_inv);
00596         gsl_matrix_scale(a, 1 + lmpar);
00597         gsl_matrix_add(a, cov);
00598
00599         /* Solve A * x_step = b by means of Cholesky decomposition... */
00600         gsl_linalg_cholesky_decomp(a);
00601         gsl_linalg_cholesky_solve(a, b, x_step);
00602
00603         /* Update atmospheric state... */
00604         gsl_vector_add(x_i, x_step);
00605         copy_atm(ctl, atm_i, atm_apr, 0);
00606         copy_obs(ctl, obs_i, obs_meas, 0);
00607         x2atm(ctl, x_i, atm_i);
00608
00609         /* Check atmospheric state... */
00610         for (ip = 0; ip < atm_i->np; ip++) {
00611             atm_i->p[ip] = GSL_MIN(GSL_MAX(atm_i->p[ip], 5e-7), 5e4);
00612             atm_i->t[ip] = GSL_MIN(GSL_MAX(atm_i->t[ip], 100), 400);
00613             for (ig = 0; ig < ctl->ng; ig++)
00614                 atm_i->q[ig][ip] = GSL_MIN(GSL_MAX(atm_i->q[ig][ip], 0), 1);
00615             for (iw = 0; iw < ctl->nw; iw++)
00616                 atm_i->k[iw][ip] = GSL_MAX(atm_i->k[iw][ip], 0);
00617         }
00618         atm_i->clz = GSL_MAX(atm_i->clz, 0);
00619         atm_i->cldz = GSL_MAX(atm_i->cldz, 0.1);
00620         for (icl = 0; icl < ctl->ncl; icl++)
00621             atm_i->clk[icl] = GSL_MAX(atm_i->clk[icl], 0);
00622         atm_i->sfz = GSL_MAX(atm_i->sfz, 0);
00623         atm_i->sfp = GSL_MAX(atm_i->sfp, 0);
00624         atm_i->sft = GSL_MIN(GSL_MAX(atm_i->sft, 100), 400);
00625         for (isf = 0; isf < ctl->nsf; isf++)
00626             atm_i->sfeps[isf] = GSL_MIN(GSL_MAX(atm_i->sfeps[isf], 0), 1);
00627
00628         /* Forward calculation... */
00629         formod(ctl, atm_i, obs_i);
00630         obs2y(ctl, obs_i, y_i, NULL, NULL);
00631
00632         /* Determine dx = x_i - x_a and dy = y - F(x_i) ... */
00633         gsl_vector_memcpy(dx, x_i);
00634         gsl_vector_sub(dx, x_a);
00635         gsl_vector_memcpy(dy, y_m);
00636         gsl_vector_sub(dy, y_i);
00637
00638         /* Compute cost function... */
00639         chisq = cost_function(dx, dy, s_a_inv, sig_eps_inv);
00640
00641         /* Modify Levenberg-Marquardt parameter... */
00642         if (chisq > chisq_old) {
00643             lmpar *= 10;
00644             gsl_vector_sub(x_i, x_step);
00645         } else {

```

```

00646         lmpar /= 10;
00647         break;
00648     }
00649 }
00650
00651 /* Write info... */
00652 LOG(1, "it= %d / chi^2/m= %g", it, chisq);
00653
00654 /* Write to cost function file... */
00655 fprintf(out, "%d %g %d %d\n", it, chisq, (int) m, (int) n);
00656
00657 /* Get normalized step size in state space... */
00658 gsl_blas_ddot(x_step, b, &disq);
00659 disq /= (double) n;
00660
00661 /* Convergence test... */
00662 if ((it == 1 || it % ret->kernel_recomp == 0) && disq < ret->conv_dmin)
00663     break;
00664 }
00665
00666 /* Close cost function file... */
00667 fclose(out);
00668
00669 /* Store results... */
00670 write_atm(ret->dir, "atm_final.tab", ctl, atm_i);
00671 write_obs(ret->dir, "obs_final.tab", ctl, obs_i);
00672 write_matrix(ret->dir, "matrix_kernel.tab", ctl, k_i,
00673             atm_i, obs_i, "y", "x", "r");
00674
00675 /* -----
00676    Analysis of retrieval results...
00677    ----- */
00678
00679 /* Check if error analysis is requested... */
00680 if (ret->err_ana) {
00681
00682     /* Allocate... */
00683     auxnm = gsl_matrix_alloc(n, m);
00684     corr = gsl_matrix_alloc(n, n);
00685     gain = gsl_matrix_alloc(n, m);
00686
00687     /* Compute inverse retrieval covariance...
00688        cov^{-1} = S_a^{-1} + K_i^T * S_eps^{-1} * K_i */
00689     matrix_product(k_i, sig_eps_inv, 1, cov);
00690     gsl_matrix_add(cov, s_a_inv);
00691
00692     /* Compute retrieval covariance... */
00693     matrix_invert(cov);
00694     write_matrix(ret->dir, "matrix_cov_ret.tab", ctl, cov,
00695                 atm_i, obs_i, "x", "x", "r");
00696     write_stddev("total", ret, ctl, atm_i, cov);
00697
00698     /* Compute correlation matrix... */
00699     for (i = 0; i < n; i++)
00700         for (j = 0; j < n; j++)
00701             gsl_matrix_set(corr, i, j, gsl_matrix_get(cov, i, j)
00702                             / sqrt(gsl_matrix_get(cov, i, i))
00703                             / sqrt(gsl_matrix_get(cov, j, j)));
00704     write_matrix(ret->dir, "matrix_corr.tab", ctl, corr,
00705                 atm_i, obs_i, "x", "x", "r");
00706
00707     /* Compute gain matrix...
00708        G = cov * K^T * S_eps^{-1} */
00709     for (i = 0; i < n; i++)
00710         for (j = 0; j < m; j++)
00711             gsl_matrix_set(auxnm, i, j, gsl_matrix_get(k_i, j, i)
00712                             * POW2(gsl_vector_get(sig_eps_inv, j)));
00713     gsl_blas_dgemm(CblasNoTrans, CblasNoTrans, 1.0, cov, auxnm, 0.0, gain);
00714     write_matrix(ret->dir, "matrix_gain.tab", ctl, gain,
00715                 atm_i, obs_i, "x", "y", "c");
00716
00717     /* Compute retrieval error due to noise... */
00718     matrix_product(gain, sig_noise, 2, a);
00719     write_stddev("noise", ret, ctl, atm_i, a);
00720
00721     /* Compute retrieval error due to forward model errors... */
00722     matrix_product(gain, sig_formod, 2, a);
00723     write_stddev("formod", ret, ctl, atm_i, a);
00724
00725     /* Compute averaging kernel matrix
00726        A = G * K ... */
00727     gsl_blas_dgemm(CblasNoTrans, CblasNoTrans, 1.0, gain, k_i, 0.0, a);
00728     write_matrix(ret->dir, "matrix_avk.tab", ctl, a,
00729                 atm_i, obs_i, "x", "x", "r");
00730
00731     /* Analyze averaging kernel matrix... */
00732     analyze_avk(ret, ctl, atm_i, iqa, ipa, a);

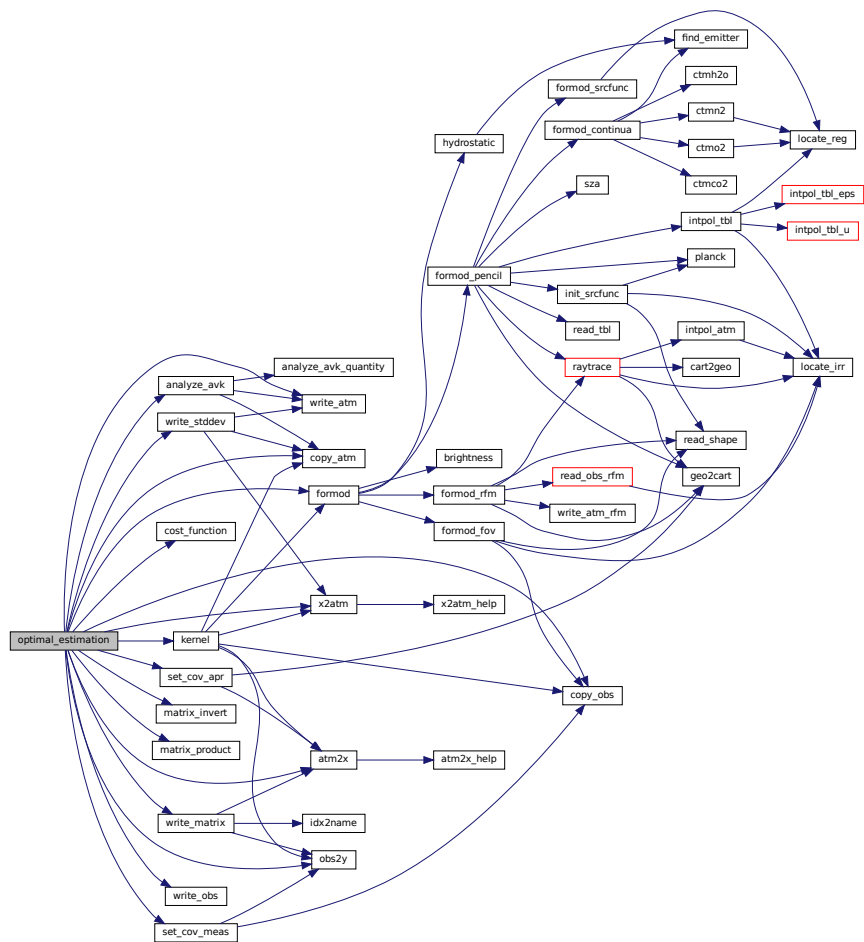
```

```

00733
00734     /* Free... */
00735     gsl_matrix_free(auxnm);
00736     gsl_matrix_free(corr);
00737     gsl_matrix_free(gain);
00738 }
00739
00740 /* -----
00741    Finalize...
00742 ----- */
00743
00744     gsl_matrix_free(a);
00745     gsl_matrix_free(cov);
00746     gsl_matrix_free(k_i);
00747     gsl_matrix_free(s_a_inv);
00748
00749     gsl_vector_free(b);
00750     gsl_vector_free(dx);
00751     gsl_vector_free(dy);
00752     gsl_vector_free(sig_eps_inv);
00753     gsl_vector_free(sig_formod);
00754     gsl_vector_free(sig_noise);
00755     gsl_vector_free(x_a);
00756     gsl_vector_free(x_i);
00757     gsl_vector_free(x_step);
00758     gsl_vector_free(y_aux);
00759     gsl_vector_free(y_i);
00760     gsl_vector_free(y_m);
00761 }

```

Here is the call graph for this function:



**5.33.2.7 read\_ret()** void read\_ret (  
     int argc,  
     char \* argv[],  
     ctl\_t \* ctl,  
     ret\_t \* ret )

Read retrieval control parameters.

Definition at line 765 of file [retrieval.c](#).

```
00769     {
00770
00771     /* Iteration control... */
00772     ret->kernel_recomp =
00773         (int) scan_ctl(argc, argv, "KERNEL_RECOMP", -1, "3", NULL);
00774     ret->conv_itmax = (int) scan_ctl(argc, argv, "CONV_ITMAX", -1, "30", NULL);
00775     ret->conv_dmin = scan_ctl(argc, argv, "CONV_DMIN", -1, "0.1", NULL);
00776
00777     /* Error analysis... */
00778     ret->err_ana = (int) scan_ctl(argc, argv, "ERR_ANA", -1, "1", NULL);
00779
00780     for (int id = 0; id < ctl->nd; id++)
00781         ret->err_formod[id] = scan_ctl(argc, argv, "ERR_FORMOD", id, "0", NULL);
00782
00783     for (int id = 0; id < ctl->nd; id++)
00784         ret->err_noise[id] = scan_ctl(argc, argv, "ERR_NOISE", id, "0", NULL);
00785
00786     ret->err_press = scan_ctl(argc, argv, "ERR_PRESS", -1, "0", NULL);
00787     ret->err_press_cz = scan_ctl(argc, argv, "ERR_PRESS_CZ", -1, "-999", NULL);
00788     ret->err_press_ch = scan_ctl(argc, argv, "ERR_PRESS_CH", -1, "-999", NULL);
00789
00790     ret->err_temp = scan_ctl(argc, argv, "ERR_TEMP", -1, "0", NULL);
00791     ret->err_temp_cz = scan_ctl(argc, argv, "ERR_TEMP_CZ", -1, "-999", NULL);
00792     ret->err_temp_ch = scan_ctl(argc, argv, "ERR_TEMP_CH", -1, "-999", NULL);
00793
00794     for (int ig = 0; ig < ctl->ng; ig++) {
00795         ret->err_q[ig] = scan_ctl(argc, argv, "ERR_Q", ig, "0", NULL);
00796         ret->err_q_cz[ig] = scan_ctl(argc, argv, "ERR_Q_CZ", ig, "-999", NULL);
00797         ret->err_q_ch[ig] = scan_ctl(argc, argv, "ERR_Q_CH", ig, "-999", NULL);
00798     }
00799
00800     for (int iw = 0; iw < ctl->nw; iw++) {
00801         ret->err_k[iw] = scan_ctl(argc, argv, "ERR_K", iw, "0", NULL);
00802         ret->err_k_cz[iw] = scan_ctl(argc, argv, "ERR_K_CZ", iw, "-999", NULL);
00803         ret->err_k_ch[iw] = scan_ctl(argc, argv, "ERR_K_CH", iw, "-999", NULL);
00804     }
00805
00806     ret->err_clz = scan_ctl(argc, argv, "ERR_CLZ", -1, "0", NULL);
00807     ret->err_cldz = scan_ctl(argc, argv, "ERR_CLDZ", -1, "0", NULL);
00808     for (int icl = 0; icl < ctl->ncl; icl++)
00809         ret->err_clk[icl] = scan_ctl(argc, argv, "ERR_CLK", icl, "0", NULL);
00810
00811     ret->err_sfz = scan_ctl(argc, argv, "ERR_SFZ", -1, "0", NULL);
00812     ret->err_sfp = scan_ctl(argc, argv, "ERR_SFP", -1, "0", NULL);
00813     ret->err_sft = scan_ctl(argc, argv, "ERR_SFT", -1, "0", NULL);
00814     for (int isf = 0; isf < ctl->nsf; isf++)
00815         ret->err_sfeps[isf] = scan_ctl(argc, argv, "ERR_SFEPS", isf, "0", NULL);
00816 }
```

Here is the call graph for this function:





```

5.33.2.8 set_cov_apr() void set_cov_apr (
    ret_t * ret,
    ctl_t * ctl,
    atm_t * atm,
    int * iqa,
    int * ipa,
    gsl_matrix * s_a )

```

Set a priori covariance.

Definition at line 820 of file [retrieval.c](#).

```

00826         {
00827
00828     gsl_vector *x_a;
00829
00830     double x0[3], x1[3];
00831
00832     /* Get sizes... */
00833     size_t n = s_a->size1;
00834
00835     /* Allocate... */
00836     x_a = gsl_vector_alloc(n);
00837
00838     /* Get sigma vector... */
00839     atm2x(ctl, atm, x_a, NULL, NULL);
00840     for (size_t i = 0; i < n; i++) {
00841         if (iqa[i] == IDXP)
00842             gsl_vector_set(x_a, i, ret->err_press / 100 * gsl_vector_get(x_a, i));
00843         if (iqa[i] == IDXT)
00844             gsl_vector_set(x_a, i, ret->err_temp);
00845         for (int ig = 0; ig < ctl->ng; ig++)
00846             if (iqa[i] == IDXQ(ig))
00847                 gsl_vector_set(x_a, i, ret->err_q[ig] / 100 * gsl_vector_get(x_a, i));
00848         for (int iw = 0; iw < ctl->nw; iw++)
00849             if (iqa[i] == IDXK(iw))
00850                 gsl_vector_set(x_a, i, ret->err_k[iw]);
00851         if (iqa[i] == IDXCLZ)
00852             gsl_vector_set(x_a, i, ret->err_clz);
00853         if (iqa[i] == IDXCLDZ)
00854             gsl_vector_set(x_a, i, ret->err_cldz);
00855         for (int icl = 0; icl < ctl->ncl; icl++)
00856             if (iqa[i] == IDXCLK(icl))
00857                 gsl_vector_set(x_a, i, ret->err_clk[icl]);
00858         if (iqa[i] == IDXSFZ)
00859             gsl_vector_set(x_a, i, ret->err_sfz);
00860         if (iqa[i] == IDXSEFP)
00861             gsl_vector_set(x_a, i, ret->err_sfp);
00862         if (iqa[i] == IDXSFST)
00863             gsl_vector_set(x_a, i, ret->err_sft);
00864         for (int isf = 0; isf < ctl->nsf; isf++)
00865             if (iqa[i] == IDXSFEPS(isf))
00866                 gsl_vector_set(x_a, i, ret->err_sfeps[isf]);
00867     }
00868
00869     /* Check standard deviations... */
00870     for (size_t i = 0; i < n; i++)
00871         if (POW2(gsl_vector_get(x_a, i)) <= 0)
00872             ERRMSG("Check a priori data (zero standard deviation)!");
00873
00874     /* Initialize diagonal covariance... */
00875     gsl_matrix_set_zero(s_a);
00876     for (size_t i = 0; i < n; i++)
00877         gsl_matrix_set(s_a, i, i, POW2(gsl_vector_get(x_a, i)));
00878
00879     /* Loop over matrix elements... */
00880     for (size_t i = 0; i < n; i++)
00881         for (size_t j = 0; j < n; j++)
00882             if (i != j && iqa[i] == iqa[j]) {
00883
00884                 /* Initialize... */
00885                 double cz = 0;
00886                 double ch = 0;
00887
00888                 /* Set correlation lengths for pressure... */
00889                 if (iqa[i] == IDXP) {
00890                     cz = ret->err_press_cz;
00891                     ch = ret->err_press_ch;
00892                 }
00893
00894                 /* Set correlation lengths for temperature... */
00895                 if (iqa[i] == IDXT) {
00896                     cz = ret->err_temp_cz;

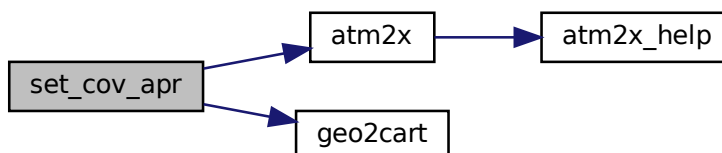
```

```

00897     ch = ret->err_temp_ch;
00898 }
00899
00900 /* Set correlation lengths for volume mixing ratios... */
00901 for (int ig = 0; ig < ctl->ng; ig++)
00902     if (iqua[i] == IDXQ(ig)) {
00903         cz = ret->err_q_cz[ig];
00904         ch = ret->err_q_ch[ig];
00905     }
00906
00907 /* Set correlation lengths for extinction... */
00908 for (int iw = 0; iw < ctl->nw; iw++)
00909     if (iqua[i] == IDXK(iw)) {
00910         cz = ret->err_k_cz[iw];
00911         ch = ret->err_k_ch[iw];
00912     }
00913
00914 /* Compute correlations... */
00915 if (cz > 0 && ch > 0) {
00916
00917     /* Get Cartesian coordinates... */
00918     geo2cart(0, atm->lon[ipa[i]], atm->lat[ipa[i]], x0);
00919     geo2cart(0, atm->lon[ipa[j]], atm->lat[ipa[j]], x1);
00920
00921     /* Compute correlations... */
00922     double rho =
00923         exp(-DIST(x0, x1) / ch -
00924             fabs(atm->z[ipa[i]] - atm->z[ipa[j]]) / cz);
00925
00926     /* Set covariance... */
00927     gsl_matrix_set(s_a, i, j, gsl_vector_get(x_a, i)
00928                 * gsl_vector_get(x_a, j) * rho);
00929 }
00930 }
00931
00932 /* Free... */
00933 gsl_vector_free(x_a);
00934 }

```

Here is the call graph for this function:



**5.33.2.9 set\_cov\_meas()** void set\_cov\_meas (

```

    ret_t * ret,
    ctl_t * ctl,
    obs_t * obs,
    gsl_vector * sig_noise,
    gsl_vector * sig_formod,
    gsl_vector * sig_eps_inv )

```

Set measurement errors.

Definition at line 938 of file [retrieval.c](#).

```

00944     {
00945

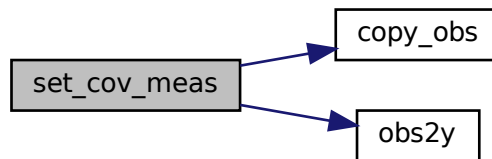
```

```

00946 static obs_t obs_err;
00947
00948 /* Get size... */
00949 size_t m = sig_eps_inv->size;
00950
00951 /* Noise error (always considered in retrieval fit)... */
00952 copy_obs(ctl, &obs_err, obs, 1);
00953 for (int ir = 0; ir < obs_err.nr; ir++)
00954     for (int id = 0; id < ctl->nd; id++)
00955         obs_err.rad[id][ir]
00956             = (gsl_finite(obs->rad[id][ir]) ? ret->err_noise[id] : GSL_NAN);
00957 obs2y(ctl, &obs_err, sig_noise, NULL, NULL);
00958
00959 /* Forward model error (always considered in retrieval fit)... */
00960 copy_obs(ctl, &obs_err, obs, 1);
00961 for (int ir = 0; ir < obs_err.nr; ir++)
00962     for (int id = 0; id < ctl->nd; id++)
00963         obs_err.rad[id][ir]
00964             = fabs(ret->err_formod[id] / 100 * obs->rad[id][ir]);
00965 obs2y(ctl, &obs_err, sig_formod, NULL, NULL);
00966
00967 /* Total error... */
00968 for (size_t i = 0; i < m; i++)
00969     gsl_vector_set(sig_eps_inv, i, 1 / sqrt(POW2(gsl_vector_get(sig_noise, i))
00970                                             +
00971                                             POW2(gsl_vector_get
00972                                                    (sig_formod, i))));
00973
00974 /* Check standard deviations... */
00975 for (size_t i = 0; i < m; i++)
00976     if (gsl_vector_get(sig_eps_inv, i) <= 0)
00977         ERRMSG("Check measurement errors (zero standard deviation)!");
00978 }

```

Here is the call graph for this function:



**5.33.2.10 write\_stddev()** void write\_stddev (
 const char \* quantity,
 ret\_t \* ret,
 ctl\_t \* ctl,
 atm\_t \* atm,
 gsl\_matrix \* s )

Write retrieval error to file.

Definition at line 982 of file [retrieval.c](#).

```

00987 {
00988
00989 static atm_t atm_aux;
00990
00991 gsl_vector *x_aux;
00992
00993 char filename[LEN];
00994

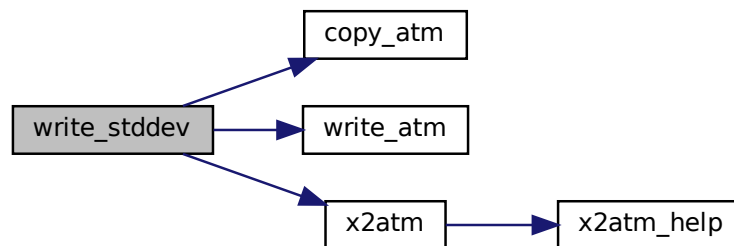
```

```

00995  /* Get sizes... */
00996  size_t n = s->size1;
00997
00998  /* Allocate... */
00999  x_aux = gsl_vector_alloc(n);
01000
01001  /* Compute standard deviation... */
01002  for (size_t i = 0; i < n; i++)
01003      gsl_vector_set(x_aux, i, sqrt(gsl_matrix_get(s, i, i)));
01004
01005  /* Write to disk... */
01006  copy_atm(ctl, &atm_aux, atm, 1);
01007  x2atm(ctl, x_aux, &atm_aux);
01008  sprintf(filename, "atm_err_%s.tab", quantity);
01009  write_atm(ret->dir, filename, ctl, &atm_aux);
01010
01011  /* Free... */
01012  gsl_vector_free(x_aux);
01013 }

```

Here is the call graph for this function:



**5.33.2.11 main()** `int main (`  
`int argc,`  
`char * argv[ ] )`

Definition at line 201 of file [retrieval.c](#).

```

00203      {
00204
00205      static atm_t atm_i, atm_apr;
00206      static ctl_t ctl;
00207      static obs_t obs_i, obs_meas;
00208      static ret_t ret;
00209
00210      FILE *dirlist;
00211
00212      /* Check arguments... */
00213      if (argc < 3)
00214          ERRMSG("Give parameters: <ctl> <dirlist>");
00215
00216      /* Measure CPU-time... */
00217      TIMER("total", 1);
00218
00219      /* Read control parameters... */
00220      read_ctl(argc, argv, &ctl);
00221      read_ret(argc, argv, &ctl, &ret);
00222
00223      /* Open directory list... */
00224      if (!(dirlist = fopen(argv[2], "r")))
00225          ERRMSG("Cannot open directory list!");
00226
00227      /* Loop over directories... */
00228      while (fscanf(dirlist, "%s", ret.dir) != EOF) {

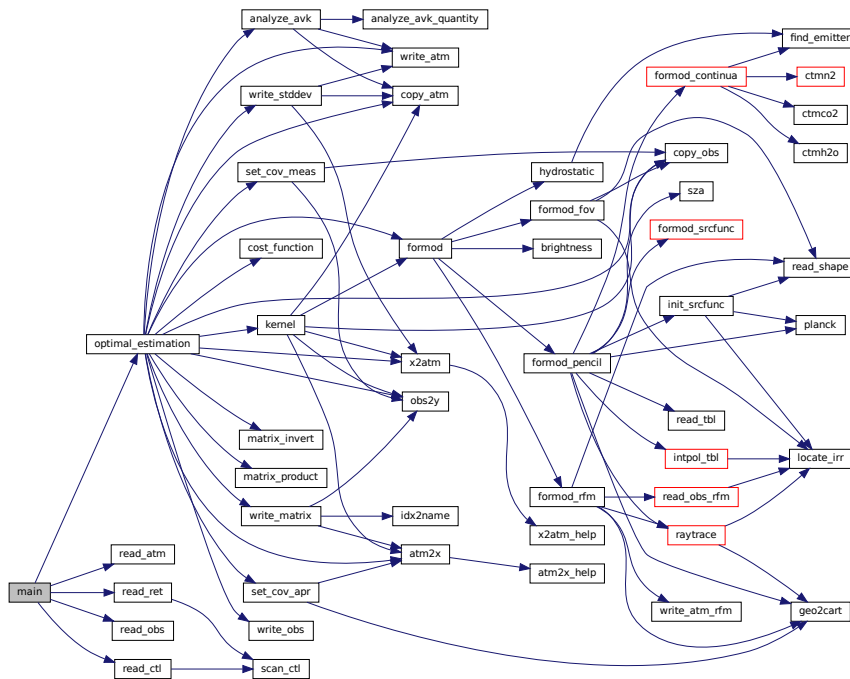
```

```

00229
00230 /* Write info... */
00231 LOG(1, "\nRetrieve in directory %s...\n", ret.dir);
00232
00233 /* Read atmospheric data... */
00234 read_atm(ret.dir, "atm_apr.tab", &ctl, &atm_apr);
00235
00236 /* Read observation data... */
00237 read_obs(ret.dir, "obs_meas.tab", &ctl, &obs_meas);
00238
00239 /* Run retrieval... */
00240 optimal_estimation(&ret, &ctl, &obs_meas, &obs_i, &atm_apr, &atm_i);
00241
00242 /* Measure CPU-time... */
00243 TIMER("total", 2);
00244 }
00245
00246 /* Write info... */
00247 LOG(1, "\nRetrieval done...");
00248
00249 /* Measure CPU-time... */
00250 TIMER("total", 3);
00251
00252 return EXIT_SUCCESS;
00253 }

```

Here is the call graph for this function:



## 5.34 retrieval.c

```

00001 /*
00002  This file is part of JURASSIC.
00003
00004  JURASSIC is free software: you can redistribute it and/or modify
00005  it under the terms of the GNU General Public License as published by
00006  the Free Software Foundation, either version 3 of the License, or
00007  (at your option) any later version.
00008
00009  JURASSIC is distributed in the hope that it will be useful,
00010  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  GNU General Public License for more details.
00013
00014  You should have received a copy of the GNU General Public License
00015  along with JURASSIC. If not, see <http://www.gnu.org/licenses/>.

```

```

00016
00017 Copyright (C) 2003-2021 Forschungszentrum Juelich GmbH
00018 */
00019
00020 #include "jurassic.h"
00021
00022 /* -----
00023   Structs...
00024   ----- */
00025
00026 typedef struct {
00027
00028   char dir[LEN];
00029
00030   int kernel_recomp;
00031
00032   int conv_itmax;
00033
00034   double conv_dmin;
00035
00036   int err_ana;
00037
00038   double err_formod[ND];
00039
00040   double err_noise[ND];
00041
00042   double err_press;
00043
00044   double err_press_cz;
00045
00046   double err_press_ch;
00047
00048   double err_temp;
00049
00050   double err_temp_cz;
00051
00052   double err_temp_ch;
00053
00054   double err_q[NG];
00055
00056   double err_q_cz[NG];
00057
00058   double err_q_ch[NG];
00059
00060   double err_k[NW];
00061
00062   double err_k_cz[NW];
00063
00064   double err_k_ch[NW];
00065
00066   double err_clz;
00067
00068   double err_cldz;
00069
00070   double err_clk[NCL];
00071
00072   double err_sfz;
00073
00074   double err_sfp;
00075
00076   double err_sft;
00077
00078   double err_sfeps[NSF];
00079 } ret_t;
00080
00081 /* -----
00082   Functions...
00083   ----- */
00084
00085 void analyze_avk(
00086   ret_t * ret,
00087   ctl_t * ctl,
00088   atm_t * atm,
00089   int *iqa,
00090   int *ipa,
00091   gsl_matrix * avk);
00092
00093 void analyze_avk_quantity(
00094   gsl_matrix * avk,
00095   int iq,
00096   int ipa,
00097   size_t *n0,
00098   size_t *n1,
00099   double *cont,
00100   double *res);
00101
00102

```

```

00138 double cost_function(
00139     gsl_vector * dx,
00140     gsl_vector * dy,
00141     gsl_matrix * s_a_inv,
00142     gsl_vector * sig_eps_inv);
00143
00145 void matrix_invert(
00146     gsl_matrix * a);
00147
00149 void matrix_product(
00150     gsl_matrix * a,
00151     gsl_vector * b,
00152     int transpose,
00153     gsl_matrix * c);
00154
00156 void optimal_estimation(
00157     ret_t * ret,
00158     ctl_t * ctl,
00159     obs_t * obs_meas,
00160     obs_t * obs_i,
00161     atm_t * atm_apr,
00162     atm_t * atm_i);
00163
00165 void read_ret(
00166     int argc,
00167     char *argv[],
00168     ctl_t * ctl,
00169     ret_t * ret);
00170
00172 void set_cov_apr(
00173     ret_t * ret,
00174     ctl_t * ctl,
00175     atm_t * atm,
00176     int *iqa,
00177     int *ipa,
00178     gsl_matrix * s_a);
00179
00181 void set_cov_meas(
00182     ret_t * ret,
00183     ctl_t * ctl,
00184     obs_t * obs,
00185     gsl_vector * sig_noise,
00186     gsl_vector * sig_formod,
00187     gsl_vector * sig_eps_inv);
00188
00190 void write_stddev(
00191     const char *quantity,
00192     ret_t * ret,
00193     ctl_t * ctl,
00194     atm_t * atm,
00195     gsl_matrix * s);
00196
00197 /* -----
00198     Main...
00199     ----- */
00200
00201 int main(
00202     int argc,
00203     char *argv[]) {
00204
00205     static atm_t atm_i, atm_apr;
00206     static ctl_t ctl;
00207     static obs_t obs_i, obs_meas;
00208     static ret_t ret;
00209
00210     FILE *dirlist;
00211
00212     /* Check arguments... */
00213     if (argc < 3)
00214         ERRMSG("Give parameters: <ctl> <dirlist>");
00215
00216     /* Measure CPU-time... */
00217     TIMER("total", 1);
00218
00219     /* Read control parameters... */
00220     read_ctl(argc, argv, &ctl);
00221     read_ret(argc, argv, &ctl, &ret);
00222
00223     /* Open directory list... */
00224     if (!(dirlist = fopen(argv[2], "r")))
00225         ERRMSG("Cannot open directory list!");
00226
00227     /* Loop over directories... */
00228     while (fscanf(dirlist, "%s", ret.dir) != EOF) {
00229
00230         /* Write info... */
00231         LOG(1, "\nRetrieve in directory %s...\n", ret.dir);

```

```

00232
00233     /* Read atmospheric data... */
00234     read_atm(ret.dir, "atm_apr.tab", &ctl, &atm_apr);
00235
00236     /* Read observation data... */
00237     read_obs(ret.dir, "obs_meas.tab", &ctl, &obs_meas);
00238
00239     /* Run retrieval... */
00240     optimal_estimation(&ret, &ctl, &obs_meas, &obs_i, &atm_apr, &atm_i);
00241
00242     /* Measure CPU-time... */
00243     TIMER("total", 2);
00244 }
00245
00246 /* Write info... */
00247 LOG(1, "\nRetrieval done...");
00248
00249 /* Measure CPU-time... */
00250 TIMER("total", 3);
00251
00252 return EXIT_SUCCESS;
00253 }
00254
00255 /*****
00256 void analyze_avk(
00257     ret_t * ret,
00258     ctl_t * ctl,
00259     atm_t * atm,
00260     int * iqa,
00261     int * ipa,
00262     gsl_matrix * avk) {
00263
00264     static atm_t atm_cont, atm_res;
00265
00266     int icl, ig, iq, isf, iw;
00267
00268     size_t i, n, n0[NQ], n1[NQ];
00269
00270     /* Get sizes... */
00271     n = avk->size1;
00272
00273     /* Find sub-matrices for different quantities... */
00274     for (iq = 0; iq < NQ; iq++) {
00275         n0[iq] = N;
00276         for (i = 0; i < n; i++) {
00277             if (iqa[i] == iq && n0[iq] == N)
00278                 n0[iq] = i;
00279             if (iqa[i] == iq)
00280                 n1[iq] = i - n0[iq] + 1;
00281         }
00282     }
00283
00284     /* Initialize... */
00285     copy_atm(ctl, &atm_cont, atm, 1);
00286     copy_atm(ctl, &atm_res, atm, 1);
00287
00288     /* Analyze quantities... */
00289     analyze_avk_quantity(avk, IDXP, ipa, n0, n1, atm_cont.p, atm_res.p);
00290     analyze_avk_quantity(avk, IDXT, ipa, n0, n1, atm_cont.t, atm_res.t);
00291     for (ig = 0; ig < ctl->ng; ig++)
00292         analyze_avk_quantity(avk, IDXQ(ig), ipa, n0, n1,
00293                             atm_cont.q[ig], atm_res.q[ig]);
00294     for (iw = 0; iw < ctl->nw; iw++)
00295         analyze_avk_quantity(avk, IDXK(iw), ipa, n0, n1,
00296                             atm_cont.k[iw], atm_res.k[iw]);
00297     analyze_avk_quantity(avk, IDXCLZ, ipa, n0, n1, &atm_cont.clz, &atm_res.clz);
00298     analyze_avk_quantity(avk, IDXCLDZ, ipa, n0, n1, &atm_cont.cldz,
00299                             &atm_res.cldz);
00300     for (icl = 0; icl < ctl->ncl; icl++)
00301         analyze_avk_quantity(avk, IDXCLK(icl), ipa, n0, n1,
00302                             &atm_cont.clk[icl], &atm_res.clk[icl]);
00303     analyze_avk_quantity(avk, IDXSFZ, ipa, n0, n1, &atm_cont.sfiz, &atm_res.sfiz);
00304     analyze_avk_quantity(avk, IDXSFP, ipa, n0, n1, &atm_cont.sfp, &atm_res.sfp);
00305     analyze_avk_quantity(avk, IDXSFZ, ipa, n0, n1, &atm_cont.sfp, &atm_res.sfp);
00306     analyze_avk_quantity(avk, IDXSFZ, ipa, n0, n1, &atm_cont.sfp, &atm_res.sfp);
00307     for (isf = 0; isf < ctl->nsf; isf++)
00308         analyze_avk_quantity(avk, IDXSFEPS(isf), ipa, n0, n1,
00309                             &atm_cont.sfeeps[isf], &atm_res.sfeeps[isf]);
00310
00311     /* Write results to disk... */
00312     write_atm(ret->dir, "atm_cont.tab", ctl, &atm_cont);
00313     write_atm(ret->dir, "atm_res.tab", ctl, &atm_res);
00314 }
00315
00316 /*****
00317 void analyze_avk_quantity(
00318

```



```

00319     gsl_matrix * avk,
00320     int iq,
00321     int *ipa,
00322     size_t *n0,
00323     size_t *n1,
00324     double *cont,
00325     double *res) {
00326
00327     /* Loop over state vector elements... */
00328     if (n0[iq] < N)
00329         for (size_t i = 0; i < n1[iq]; i++) {
00330
00331             /* Get area of averaging kernel... */
00332             for (size_t j = 0; j < n1[iq]; j++)
00333                 cont[ipa[n0[iq] + i]] += gsl_matrix_get(avk, n0[iq] + i, n0[iq] + j);
00334
00335             /* Get information density... */
00336             res[ipa[n0[iq] + i]] = 1 / gsl_matrix_get(avk, n0[iq] + i, n0[iq] + i);
00337         }
00338 }
00339
00340 /*****
00341
00342 double cost_function(
00343     gsl_vector * dx,
00344     gsl_vector * dy,
00345     gsl_matrix * s_a_inv,
00346     gsl_vector * sig_eps_inv) {
00347
00348     gsl_vector *x_aux, *y_aux;
00349
00350     double chisq_a, chisq_m = 0;
00351
00352     /* Get sizes... */
00353     size_t m = dy->size;
00354     size_t n = dx->size;
00355
00356     /* Allocate... */
00357     x_aux = gsl_vector_alloc(n);
00358     y_aux = gsl_vector_alloc(m);
00359
00360     /* Determine normalized cost function...
00361     (chi^2 = 1/m * [dy^T * S_eps^{-1} * dy + dx^T * S_a^{-1} * dx]) */
00362     for (size_t i = 0; i < m; i++)
00363         chisq_m += POW2(gsl_vector_get(dy, i) * gsl_vector_get(sig_eps_inv, i));
00364     gsl_blas_dgemv(CblasNoTrans, 1.0, s_a_inv, dx, 0.0, x_aux);
00365     gsl_blas_ddot(dx, x_aux, &chisq_a);
00366
00367     /* Free... */
00368     gsl_vector_free(x_aux);
00369     gsl_vector_free(y_aux);
00370
00371     /* Return cost function value... */
00372     return (chisq_m + chisq_a) / (double) m;
00373 }
00374
00375 /*****
00376
00377 void matrix_invert(
00378     gsl_matrix * a) {
00379
00380     size_t diag = 1;
00381
00382     /* Get size... */
00383     size_t n = a->size1;
00384
00385     /* Check if matrix is diagonal... */
00386     for (size_t i = 0; i < n && diag; i++)
00387         for (size_t j = i + 1; j < n; j++)
00388             if (gsl_matrix_get(a, i, j) != 0) {
00389                 diag = 0;
00390                 break;
00391             }
00392
00393     /* Quick inversion of diagonal matrix... */
00394     if (diag)
00395         for (size_t i = 0; i < n; i++)
00396             gsl_matrix_set(a, i, i, 1 / gsl_matrix_get(a, i, i));
00397
00398     /* Matrix inversion by means of Cholesky decomposition... */
00399     else {
00400         gsl_linalg_cholesky_decomp(a);
00401         gsl_linalg_cholesky_invert(a);
00402     }
00403 }
00404
00405 /*****

```

```

00406
00407 void matrix_product(
00408     gsl_matrix * a,
00409     gsl_vector * b,
00410     int transpose,
00411     gsl_matrix * c) {
00412
00413     gsl_matrix *aux;
00414
00415     /* Set sizes... */
00416     size_t m = a->size1;
00417     size_t n = a->size2;
00418
00419     /* Allocate... */
00420     aux = gsl_matrix_alloc(m, n);
00421
00422     /* Compute A^T B A... */
00423     if (transpose == 1) {
00424
00425         /* Compute B^1/2 A... */
00426         for (size_t i = 0; i < m; i++)
00427             for (size_t j = 0; j < n; j++)
00428                 gsl_matrix_set(aux, i, j,
00429                     gsl_vector_get(b, i) * gsl_matrix_get(a, i, j));
00430
00431         /* Compute A^T B A = (B^1/2 A)^T (B^1/2 A)... */
00432         gsl_blas_dgemm(CblasTrans, CblasNoTrans, 1.0, aux, aux, 0.0, c);
00433     }
00434
00435     /* Compute A B A^T... */
00436     else if (transpose == 2) {
00437
00438         /* Compute A B^1/2... */
00439         for (size_t i = 0; i < m; i++)
00440             for (size_t j = 0; j < n; j++)
00441                 gsl_matrix_set(aux, i, j,
00442                     gsl_matrix_get(a, i, j) * gsl_vector_get(b, j));
00443
00444         /* Compute A B A^T = (A B^1/2) (A B^1/2)^T... */
00445         gsl_blas_dgemm(CblasNoTrans, CblasTrans, 1.0, aux, aux, 0.0, c);
00446     }
00447
00448     /* Free... */
00449     gsl_matrix_free(aux);
00450 }
00451
00452 /*****
00453
00454 void optimal_estimation(
00455     ret_t * ret,
00456     ctl_t * ctl,
00457     obs_t * obs_meas,
00458     obs_t * obs_i,
00459     atm_t * atm_apr,
00460     atm_t * atm_i) {
00461
00462     static int ipa[N], iqa[N];
00463
00464     gsl_matrix *a, *auxnm, *corr, *cov, *gain, *k_i, *s_a_inv;
00465     gsl_vector *b, *dx, *dy, *sig_eps_inv, *sig_formod, *sig_noise,
00466         *x_a, *x_i, *x_step, *y_aux, *y_i, *y_m;
00467
00468     FILE *out;
00469
00470     char filename[2 * LEN];
00471
00472     double chisq, chisq_old, disq = 0, lmpar = 0.001;
00473
00474     int icl, ig, ip, isf, it = 0, it2, iw;
00475
00476     size_t i, j, m, n;
00477
00478     /* -----
00479        Initialize...
00480        ----- */
00481
00482     /* Get sizes... */
00483     m = obs2y(ctl, obs_meas, NULL, NULL, NULL);
00484     n = atm2x(ctl, atm_apr, NULL, iqa, ipa);
00485     if (m == 0 || n == 0)
00486         ERRMSG("Check problem definition!");
00487
00488     /* Write info... */
00489     LOG(1, "Problem size: m= %d / n= %d "
00490         "(alloc= %.4g MB / stat= %.4g MB)",
00491         (int) m, (int) n,
00492         (double) (3 * m * n + 4 * n * n + 8 * m +

```

```

00493         8 * n) * sizeof(double) / 1024. / 1024.,
00494         (double) (5 * sizeof(atm_t) + 3 * sizeof(obs_t)
00495         + 2 * N * sizeof(int)) / 1024. / 1024.);
00496
00497     /* Allocate... */
00498     a = gsl_matrix_alloc(n, n);
00499     cov = gsl_matrix_alloc(n, n);
00500     k_i = gsl_matrix_alloc(m, n);
00501     s_a_inv = gsl_matrix_alloc(n, n);
00502
00503     b = gsl_vector_alloc(n);
00504     dx = gsl_vector_alloc(n);
00505     dy = gsl_vector_alloc(m);
00506     sig_eps_inv = gsl_vector_alloc(m);
00507     sig_formod = gsl_vector_alloc(m);
00508     sig_noise = gsl_vector_alloc(m);
00509     x_a = gsl_vector_alloc(n);
00510     x_i = gsl_vector_alloc(n);
00511     x_step = gsl_vector_alloc(n);
00512     y_aux = gsl_vector_alloc(m);
00513     y_i = gsl_vector_alloc(m);
00514     y_m = gsl_vector_alloc(m);
00515
00516     /* Set initial state... */
00517     copy_atm(ctl, atm_i, atm_apr, 0);
00518     copy_obs(ctl, obs_i, obs_meas, 0);
00519     formod(ctl, atm_i, obs_i);
00520
00521     /* Set state vectors and observation vectors... */
00522     atm2x(ctl, atm_apr, x_a, NULL, NULL);
00523     atm2x(ctl, atm_i, x_i, NULL, NULL);
00524     obs2y(ctl, obs_meas, y_m, NULL, NULL);
00525     obs2y(ctl, obs_i, y_i, NULL, NULL);
00526
00527     /* Set inverse a priori covariance S_a^-1... */
00528     set_cov_apr(ret, ctl, atm_apr, iqa, ipa, s_a_inv);
00529     write_matrix(ret->dir, "matrix_cov_apr.tab", ctl, s_a_inv,
00530                 atm_i, obs_i, "x", "x", "r");
00531     matrix_invert(s_a_inv);
00532
00533     /* Get measurement errors... */
00534     set_cov_meas(ret, ctl, obs_meas, sig_noise, sig_formod, sig_eps_inv);
00535
00536     /* Create cost function file... */
00537     sprintf(filename, "%s/costs.tab", ret->dir);
00538     if (!(out = fopen(filename, "w")))
00539         ERRMSG("Cannot create cost function file!");
00540
00541     /* Write header... */
00542     fprintf(out,
00543            "# $1 = iteration number\n"
00544            "# $2 = normalized cost function\n"
00545            "# $3 = number of measurements\n"
00546            "# $4 = number of state vector elements\n\n");
00547
00548     /* Determine dx = x_i - x_a and dy = y - F(x_i) ... */
00549     gsl_vector_memcpy(dx, x_i);
00550     gsl_vector_sub(dx, x_a);
00551     gsl_vector_memcpy(dy, y_m);
00552     gsl_vector_sub(dy, y_i);
00553
00554     /* Compute cost function... */
00555     chisq = cost_function(dx, dy, s_a_inv, sig_eps_inv);
00556
00557     /* Write info... */
00558     LOG(1, "it= %d / chi^2/m= %g", it, chisq);
00559
00560     /* Write to cost function file... */
00561     fprintf(out, "%d %g %d %d\n", it, chisq, (int) m, (int) n);
00562
00563     /* Compute initial kernel... */
00564     kernel(ctl, atm_i, obs_i, k_i);
00565
00566     /* -----
00567     Levenberg-Marquardt minimization...
00568     ----- */
00569
00570     /* Outer loop... */
00571     for (it = 1; it <= ret->conv_itmax; it++) {
00572
00573         /* Store current cost function value... */
00574         chisq_old = chisq;
00575
00576         /* Compute kernel matrix K_i... */
00577         if (it > 1 && it % ret->kernel_recomp == 0)
00578             kernel(ctl, atm_i, obs_i, k_i);
00579

```

```

00580      /* Compute  $K_i^T * S_{\text{eps}}^{-1} * K_i$  ... */
00581      if (it == 1 || it % ret->kernel_recomp == 0)
00582          matrix_product(k_i, sig_eps_inv, 1, cov);
00583
00584      /* Determine  $b = K_i^T * S_{\text{eps}}^{-1} * dy - S_a^{-1} * dx$  ... */
00585      for (i = 0; i < m; i++)
00586          gsl_vector_set(y_aux, i, gsl_vector_get(dy, i)
00587                          * POW2(gsl_vector_get(sig_eps_inv, i)));
00588      gsl_blas_dgemv(CblasTrans, 1.0, k_i, y_aux, 0.0, b);
00589      gsl_blas_dgemv(CblasNoTrans, -1.0, s_a_inv, dx, 1.0, b);
00590
00591      /* Inner loop... */
00592      for (it2 = 0; it2 < 20; it2++) {
00593
00594          /* Compute  $A = (1 + \text{lmpar}) * S_a^{-1} + K_i^T * S_{\text{eps}}^{-1} * K_i$  ... */
00595          gsl_matrix_memcpy(a, s_a_inv);
00596          gsl_matrix_scale(a, 1 + lmpar);
00597          gsl_matrix_add(a, cov);
00598
00599          /* Solve  $A * x_{\text{step}} = b$  by means of Cholesky decomposition... */
00600          gsl_linalg_cholesky_decomp(a);
00601          gsl_linalg_cholesky_solve(a, b, x_step);
00602
00603          /* Update atmospheric state... */
00604          gsl_vector_add(x_i, x_step);
00605          copy_atm(ctl, atm_i, atm_apr, 0);
00606          copy_obs(ctl, obs_i, obs_meas, 0);
00607          x2atm(ctl, x_i, atm_i);
00608
00609          /* Check atmospheric state... */
00610          for (ip = 0; ip < atm_i->np; ip++) {
00611              atm_i->p[ip] = GSL_MIN(GSL_MAX(atm_i->p[ip], 5e-7), 5e4);
00612              atm_i->t[ip] = GSL_MIN(GSL_MAX(atm_i->t[ip], 100), 400);
00613              for (ig = 0; ig < ctl->ng; ig++)
00614                  atm_i->q[ig][ip] = GSL_MIN(GSL_MAX(atm_i->q[ig][ip], 0), 1);
00615              for (iw = 0; iw < ctl->nw; iw++)
00616                  atm_i->k[iw][ip] = GSL_MAX(atm_i->k[iw][ip], 0);
00617          }
00618          atm_i->clz = GSL_MAX(atm_i->clz, 0);
00619          atm_i->cldz = GSL_MAX(atm_i->cldz, 0.1);
00620          for (icl = 0; icl < ctl->ncl; icl++)
00621              atm_i->clk[icl] = GSL_MAX(atm_i->clk[icl], 0);
00622          atm_i->sfz = GSL_MAX(atm_i->sfz, 0);
00623          atm_i->sfp = GSL_MAX(atm_i->sfp, 0);
00624          atm_i->sft = GSL_MIN(GSL_MAX(atm_i->sft, 100), 400);
00625          for (isf = 0; isf < ctl->nsf; isf++)
00626              atm_i->sfeps[isf] = GSL_MIN(GSL_MAX(atm_i->sfeps[isf], 0), 1);
00627
00628          /* Forward calculation... */
00629          formod(ctl, atm_i, obs_i);
00630          obs2y(ctl, obs_i, y_i, NULL, NULL);
00631
00632          /* Determine  $dx = x_i - x_a$  and  $dy = y - F(x_i)$  ... */
00633          gsl_vector_memcpy(dx, x_i);
00634          gsl_vector_sub(dx, x_a);
00635          gsl_vector_memcpy(dy, y_m);
00636          gsl_vector_sub(dy, y_i);
00637
00638          /* Compute cost function... */
00639          chisq = cost_function(dx, dy, s_a_inv, sig_eps_inv);
00640
00641          /* Modify Levenberg-Marquardt parameter... */
00642          if (chisq > chisq_old) {
00643              lmpar *= 10;
00644              gsl_vector_sub(x_i, x_step);
00645          } else {
00646              lmpar /= 10;
00647              break;
00648          }
00649      }
00650
00651      /* Write info... */
00652      LOG(1, "it= %d / chi^2/m= %g", it, chisq);
00653
00654      /* Write to cost function file... */
00655      fprintf(out, "%d %g %d %d\n", it, chisq, (int) m, (int) n);
00656
00657      /* Get normalized step size in state space... */
00658      gsl_blas_ddot(x_step, b, &disq);
00659      disq /= (double) n;
00660
00661      /* Convergence test... */
00662      if ((it == 1 || it % ret->kernel_recomp == 0) && disq < ret->conv_dmin)
00663          break;
00664  }
00665
00666      /* Close cost function file... */

```

```

00667     fclose(out);
00668
00669     /* Store results... */
00670     write_atm(ret->dir, "atm_final.tab", ctl, atm_i);
00671     write_obs(ret->dir, "obs_final.tab", ctl, obs_i);
00672     write_matrix(ret->dir, "matrix_kernel.tab", ctl, k_i,
00673                 atm_i, obs_i, "y", "x", "r");
00674
00675     /* -----
00676        Analysis of retrieval results...
00677        ----- */
00678
00679     /* Check if error analysis is requested... */
00680     if (ret->err_ana) {
00681
00682         /* Allocate... */
00683         auxnm = gsl_matrix_alloc(n, m);
00684         corr = gsl_matrix_alloc(n, n);
00685         gain = gsl_matrix_alloc(n, m);
00686
00687         /* Compute inverse retrieval covariance...
00688             $\text{cov}^{-1} = \text{S}_a^{-1} + \text{K}_i^T * \text{S}_{\text{eps}}^{-1} * \text{K}_i$  */
00689         matrix_product(k_i, sig_eps_inv, 1, cov);
00690         gsl_matrix_add(cov, s_a_inv);
00691
00692         /* Compute retrieval covariance... */
00693         matrix_invert(cov);
00694         write_matrix(ret->dir, "matrix_cov_ret.tab", ctl, cov,
00695                     atm_i, obs_i, "x", "x", "r");
00696         write_stddev("total", ret, ctl, atm_i, cov);
00697
00698         /* Compute correlation matrix... */
00699         for (i = 0; i < n; i++)
00700             for (j = 0; j < n; j++)
00701                 gsl_matrix_set(corr, i, j, gsl_matrix_get(cov, i, j)
00702                               / sqrt(gsl_matrix_get(cov, i, i))
00703                               / sqrt(gsl_matrix_get(cov, j, j)));
00704         write_matrix(ret->dir, "matrix_corr.tab", ctl, corr,
00705                     atm_i, obs_i, "x", "x", "r");
00706
00707         /* Compute gain matrix...
00708             $G = \text{cov} * \text{K}^T * \text{S}_{\text{eps}}^{-1}$  */
00709         for (i = 0; i < n; i++)
00710             for (j = 0; j < m; j++)
00711                 gsl_matrix_set(auxnm, i, j, gsl_matrix_get(k_i, j, i)
00712                               * POW2(gsl_vector_get(sig_eps_inv, j)));
00713         gsl_blas_dgemm(CblasNoTrans, CblasNoTrans, 1.0, cov, auxnm, 0.0, gain);
00714         write_matrix(ret->dir, "matrix_gain.tab", ctl, gain,
00715                     atm_i, obs_i, "x", "y", "c");
00716
00717         /* Compute retrieval error due to noise... */
00718         matrix_product(gain, sig_noise, 2, a);
00719         write_stddev("noise", ret, ctl, atm_i, a);
00720
00721         /* Compute retrieval error due to forward model errors... */
00722         matrix_product(gain, sig_formod, 2, a);
00723         write_stddev("formod", ret, ctl, atm_i, a);
00724
00725         /* Compute averaging kernel matrix
00726             $A = G * K \dots$  */
00727         gsl_blas_dgemm(CblasNoTrans, CblasNoTrans, 1.0, gain, k_i, 0.0, a);
00728         write_matrix(ret->dir, "matrix_avk.tab", ctl, a,
00729                     atm_i, obs_i, "x", "x", "r");
00730
00731         /* Analyze averaging kernel matrix... */
00732         analyze_avk(ret, ctl, atm_i, iqa, ipa, a);
00733
00734         /* Free... */
00735         gsl_matrix_free(auxnm);
00736         gsl_matrix_free(corr);
00737         gsl_matrix_free(gain);
00738     }
00739
00740     /* -----
00741        Finalize...
00742        ----- */
00743
00744     gsl_matrix_free(a);
00745     gsl_matrix_free(cov);
00746     gsl_matrix_free(k_i);
00747     gsl_matrix_free(s_a_inv);
00748
00749     gsl_vector_free(b);
00750     gsl_vector_free(dx);
00751     gsl_vector_free(dy);
00752     gsl_vector_free(sig_eps_inv);
00753     gsl_vector_free(sig_formod);

```

```

00754     gsl_vector_free(sig_noise);
00755     gsl_vector_free(x_a);
00756     gsl_vector_free(x_i);
00757     gsl_vector_free(x_step);
00758     gsl_vector_free(y_aux);
00759     gsl_vector_free(y_i);
00760     gsl_vector_free(y_m);
00761 }
00762
00763 /*****
00764
00765 void read_ret(
00766     int argc,
00767     char *argv[],
00768     ctl_t * ctl,
00769     ret_t * ret) {
00770
00771     /* Iteration control... */
00772     ret->kernel_recomp =
00773         (int) scan_ctl(argc, argv, "KERNEL_RECOMP", -1, "3", NULL);
00774     ret->conv_itmax = (int) scan_ctl(argc, argv, "CONV_ITMAX", -1, "30", NULL);
00775     ret->conv_dmin = scan_ctl(argc, argv, "CONV_DMIN", -1, "0.1", NULL);
00776
00777     /* Error analysis... */
00778     ret->err_ana = (int) scan_ctl(argc, argv, "ERR_ANA", -1, "1", NULL);
00779
00780     for (int id = 0; id < ctl->nd; id++)
00781         ret->err_formod[id] = scan_ctl(argc, argv, "ERR_FORMOD", id, "0", NULL);
00782
00783     for (int id = 0; id < ctl->nd; id++)
00784         ret->err_noise[id] = scan_ctl(argc, argv, "ERR_NOISE", id, "0", NULL);
00785
00786     ret->err_press = scan_ctl(argc, argv, "ERR_PRESS", -1, "0", NULL);
00787     ret->err_press_cz = scan_ctl(argc, argv, "ERR_PRESS_CZ", -1, "-999", NULL);
00788     ret->err_press_ch = scan_ctl(argc, argv, "ERR_PRESS_CH", -1, "-999", NULL);
00789
00790     ret->err_temp = scan_ctl(argc, argv, "ERR_TEMP", -1, "0", NULL);
00791     ret->err_temp_cz = scan_ctl(argc, argv, "ERR_TEMP_CZ", -1, "-999", NULL);
00792     ret->err_temp_ch = scan_ctl(argc, argv, "ERR_TEMP_CH", -1, "-999", NULL);
00793
00794     for (int ig = 0; ig < ctl->ng; ig++) {
00795         ret->err_q[ig] = scan_ctl(argc, argv, "ERR_Q", ig, "0", NULL);
00796         ret->err_q_cz[ig] = scan_ctl(argc, argv, "ERR_Q_CZ", ig, "-999", NULL);
00797         ret->err_q_ch[ig] = scan_ctl(argc, argv, "ERR_Q_CH", ig, "-999", NULL);
00798     }
00799
00800     for (int iw = 0; iw < ctl->nw; iw++) {
00801         ret->err_k[iw] = scan_ctl(argc, argv, "ERR_K", iw, "0", NULL);
00802         ret->err_k_cz[iw] = scan_ctl(argc, argv, "ERR_K_CZ", iw, "-999", NULL);
00803         ret->err_k_ch[iw] = scan_ctl(argc, argv, "ERR_K_CH", iw, "-999", NULL);
00804     }
00805
00806     ret->err_clz = scan_ctl(argc, argv, "ERR_CLZ", -1, "0", NULL);
00807     ret->err_cldz = scan_ctl(argc, argv, "ERR_CLDZ", -1, "0", NULL);
00808     for (int icl = 0; icl < ctl->ncl; icl++)
00809         ret->err_clk[icl] = scan_ctl(argc, argv, "ERR_CLK", icl, "0", NULL);
00810
00811     ret->err_sfz = scan_ctl(argc, argv, "ERR_SFZ", -1, "0", NULL);
00812     ret->err_sfp = scan_ctl(argc, argv, "ERR_SFP", -1, "0", NULL);
00813     ret->err_sft = scan_ctl(argc, argv, "ERR_SFT", -1, "0", NULL);
00814     for (int isf = 0; isf < ctl->nsf; isf++)
00815         ret->err_sfeps[isf] = scan_ctl(argc, argv, "ERR_SFEPS", isf, "0", NULL);
00816 }
00817
00818 /*****
00819
00820 void set_cov_apr(
00821     ret_t * ret,
00822     ctl_t * ctl,
00823     atm_t * atm,
00824     int *iqa,
00825     int *ipa,
00826     gsl_matrix * s_a) {
00827
00828     gsl_vector *x_a;
00829
00830     double x0[3], x1[3];
00831
00832     /* Get sizes... */
00833     size_t n = s_a->size1;
00834
00835     /* Allocate... */
00836     x_a = gsl_vector_alloc(n);
00837
00838     /* Get sigma vector... */
00839     atm2x(ctl, atm, x_a, NULL, NULL);
00840     for (size_t i = 0; i < n; i++) {

```

```

00841     if (iqa[i] == IDXP)
00842         gsl_vector_set(x_a, i, ret->err_press / 100 * gsl_vector_get(x_a, i));
00843     if (iqa[i] == IDXT)
00844         gsl_vector_set(x_a, i, ret->err_temp);
00845     for (int ig = 0; ig < ctl->ng; ig++)
00846         if (iqa[i] == IDXQ(ig))
00847             gsl_vector_set(x_a, i, ret->err_q[ig] / 100 * gsl_vector_get(x_a, i));
00848     for (int iw = 0; iw < ctl->nw; iw++)
00849         if (iqa[i] == IDXK(iw))
00850             gsl_vector_set(x_a, i, ret->err_k[iw]);
00851     if (iqa[i] == IDXCLZ)
00852         gsl_vector_set(x_a, i, ret->err_clz);
00853     if (iqa[i] == IDXCILDZ)
00854         gsl_vector_set(x_a, i, ret->err_cldz);
00855     for (int icl = 0; icl < ctl->ncl; icl++)
00856         if (iqa[i] == IDXCLK(icl))
00857             gsl_vector_set(x_a, i, ret->err_clk[icl]);
00858     if (iqa[i] == IDXSFPZ)
00859         gsl_vector_set(x_a, i, ret->err_sfz);
00860     if (iqa[i] == IDXSFP)
00861         gsl_vector_set(x_a, i, ret->err_sfp);
00862     if (iqa[i] == IDXSFT)
00863         gsl_vector_set(x_a, i, ret->err_sft);
00864     for (int isf = 0; isf < ctl->nsf; isf++)
00865         if (iqa[i] == IDXSFEPS(isf))
00866             gsl_vector_set(x_a, i, ret->err_sfeps[isf]);
00867 }
00868
00869 /* Check standard deviations... */
00870 for (size_t i = 0; i < n; i++)
00871     if (POW2(gsl_vector_get(x_a, i)) <= 0)
00872         ERRMSG("Check a priori data (zero standard deviation)!");
00873
00874 /* Initialize diagonal covariance... */
00875 gsl_matrix_set_zero(s_a);
00876 for (size_t i = 0; i < n; i++)
00877     gsl_matrix_set(s_a, i, i, POW2(gsl_vector_get(x_a, i)));
00878
00879 /* Loop over matrix elements... */
00880 for (size_t i = 0; i < n; i++)
00881     for (size_t j = 0; j < n; j++)
00882         if (i != j && iqa[i] == iqa[j]) {
00883
00884             /* Initialize... */
00885             double cz = 0;
00886             double ch = 0;
00887
00888             /* Set correlation lengths for pressure... */
00889             if (iqa[i] == IDXP) {
00890                 cz = ret->err_press_cz;
00891                 ch = ret->err_press_ch;
00892             }
00893
00894             /* Set correlation lengths for temperature... */
00895             if (iqa[i] == IDXT) {
00896                 cz = ret->err_temp_cz;
00897                 ch = ret->err_temp_ch;
00898             }
00899
00900             /* Set correlation lengths for volume mixing ratios... */
00901             for (int ig = 0; ig < ctl->ng; ig++)
00902                 if (iqa[i] == IDXQ(ig)) {
00903                     cz = ret->err_q_cz[ig];
00904                     ch = ret->err_q_ch[ig];
00905                 }
00906
00907             /* Set correlation lengths for extinction... */
00908             for (int iw = 0; iw < ctl->nw; iw++)
00909                 if (iqa[i] == IDXK(iw)) {
00910                     cz = ret->err_k_cz[iw];
00911                     ch = ret->err_k_ch[iw];
00912                 }
00913
00914             /* Compute correlations... */
00915             if (cz > 0 && ch > 0) {
00916
00917                 /* Get Cartesian coordinates... */
00918                 geo2cart(0, atm->lon[ipa[i]], atm->lat[ipa[i]], x0);
00919                 geo2cart(0, atm->lon[ipa[j]], atm->lat[ipa[j]], x1);
00920
00921                 /* Compute correlations... */
00922                 double rho =
00923                     exp(-DIST(x0, x1) / ch -
00924                        fabs(atm->z[ipa[i]] - atm->z[ipa[j]]) / cz);
00925
00926                 /* Set covariance... */
00927                 gsl_matrix_set(s_a, i, j, gsl_vector_get(x_a, i)

```

```

00928             * gsl_vector_get(x_a, j) * rho;
00929         }
00930     }
00931
00932     /* Free... */
00933     gsl_vector_free(x_a);
00934 }
00935
00936 /*****
00937
00938 void set_cov_meas(
00939     ret_t * ret,
00940     ctl_t * ctl,
00941     obs_t * obs,
00942     gsl_vector * sig_noise,
00943     gsl_vector * sig_formod,
00944     gsl_vector * sig_eps_inv) {
00945
00946     static obs_t obs_err;
00947
00948     /* Get size... */
00949     size_t m = sig_eps_inv->size;
00950
00951     /* Noise error (always considered in retrieval fit)... */
00952     copy_obs(ctl, &obs_err, obs, 1);
00953     for (int ir = 0; ir < obs_err.nr; ir++)
00954         for (int id = 0; id < ctl->nd; id++)
00955             obs_err.rad[id][ir]
00956                 = (gsl_finite(obs->rad[id][ir]) ? ret->err_noise[id] : GSL_NAN);
00957     obs2y(ctl, &obs_err, sig_noise, NULL, NULL);
00958
00959     /* Forward model error (always considered in retrieval fit)... */
00960     copy_obs(ctl, &obs_err, obs, 1);
00961     for (int ir = 0; ir < obs_err.nr; ir++)
00962         for (int id = 0; id < ctl->nd; id++)
00963             obs_err.rad[id][ir]
00964                 = fabs(ret->err_formod[id] / 100 * obs->rad[id][ir]);
00965     obs2y(ctl, &obs_err, sig_formod, NULL, NULL);
00966
00967     /* Total error... */
00968     for (size_t i = 0; i < m; i++)
00969         gsl_vector_set(sig_eps_inv, i, 1 / sqrt(POW2(gsl_vector_get(sig_noise, i))
00970             +
00971             POW2(gsl_vector_get
00972                 (sig_formod, i))));
00973
00974     /* Check standard deviations... */
00975     for (size_t i = 0; i < m; i++)
00976         if (gsl_vector_get(sig_eps_inv, i) <= 0)
00977             ERRMSG("Check measurement errors (zero standard deviation)!");
00978 }
00979
00980 /*****
00981
00982 void write_stddev(
00983     const char *quantity,
00984     ret_t * ret,
00985     ctl_t * ctl,
00986     atm_t * atm,
00987     gsl_matrix * s) {
00988
00989     static atm_t atm_aux;
00990
00991     gsl_vector *x_aux;
00992
00993     char filename[LEN];
00994
00995     /* Get sizes... */
00996     size_t n = s->size1;
00997
00998     /* Allocate... */
00999     x_aux = gsl_vector_alloc(n);
01000
01001     /* Compute standard deviation... */
01002     for (size_t i = 0; i < n; i++)
01003         gsl_vector_set(x_aux, i, sqrt(gsl_matrix_get(s, i, i)));
01004
01005     /* Write to disk... */
01006     copy_atm(ctl, &atm_aux, atm, 1);
01007     x2atm(ctl, x_aux, &atm_aux);
01008     sprintf(filename, "atm_err_%s.tab", quantity);
01009     write_atm(ret->dir, filename, ctl, &atm_aux);
01010
01011     /* Free... */
01012     gsl_vector_free(x_aux);
01013 }

```



## 5.35 tblfmt.c File Reference

Convert look-up table file format.

```
#include "jurassic.h"
```

### Functions

- [int main](#) (int argc, char \*argv[])

#### 5.35.1 Detailed Description

Convert look-up table file format.

Definition in file [tblfmt.c](#).

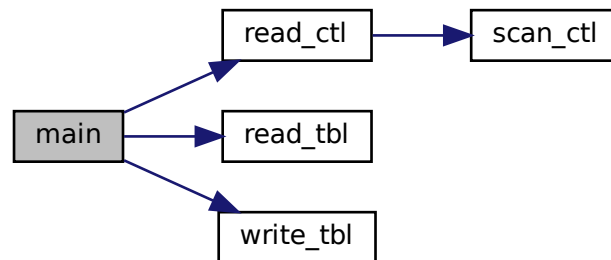
#### 5.35.2 Function Documentation

**5.35.2.1 main()** `int main (`  
     `int argc,`  
     `char * argv[] )`

Definition at line 27 of file [tblfmt.c](#).

```
00029     {
00030
00031     ctl_t ctl;
00032
00033     tbl_t *tbl;
00034
00035     /* Check arguments... */
00036     if (argc < 6)
00037         ERRMSG("Give parameters: <ctl> <tblbase_in> <tblfmt_in>"
00038              " <tblbase_out> <tblfmt_out>");
00039
00040     /* Read control parameters... */
00041     read_ctl(argc, argv, &ctl);
00042
00043     /* Allocate... */
00044     ALLOC(tbl, tbl_t, 1);
00045
00046     /* Read tables... */
00047     sprintf(ctl.tblbase, "%s", argv[2]);
00048     ctl.tblfmt = atoi(argv[3]);
00049     read_tbl(&ctl, tbl);
00050
00051     /* Write tables... */
00052     sprintf(ctl.tblbase, "%s", argv[4]);
00053     ctl.tblfmt = atoi(argv[5]);
00054     write_tbl(&ctl, tbl);
00055
00056     /* Free... */
00057     free(tbl);
00058
00059     return EXIT_SUCCESS;
00060 }
```

Here is the call graph for this function:



## 5.36 tblfmt.c

```

00001 /*
00002  This file is part of JURASSIC.
00003
00004  JURASSIC is free software: you can redistribute it and/or modify
00005  it under the terms of the GNU General Public License as published by
00006  the Free Software Foundation, either version 3 of the License, or
00007  (at your option) any later version.
00008
00009  JURASSIC is distributed in the hope that it will be useful,
00010  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  GNU General Public License for more details.
00013
00014  You should have received a copy of the GNU General Public License
00015  along with JURASSIC. If not, see <http://www.gnu.org/licenses/>.
00016
00017  Copyright (C) 2013-2021 Forschungszentrum Juelich GmbH
00018 */
00019
00025 #include "jurassic.h"
00026
00027 int main(
00028     int argc,
00029     char *argv[]) {
00030
00031     ctl_t ctl;
00032
00033     tbl_t *tbl;
00034
00035     /* Check arguments... */
00036     if (argc < 6)
00037         ERRMSG("Give parameters: <ctl> <tblbase_in> <tblfmt_in>"
00038             " <tblbase_out> <tblfmt_out>");
00039
00040     /* Read control parameters... */
00041     read_ctl(argc, argv, &ctl);
00042
00043     /* Allocate... */
00044     ALLOC(tbl, tbl_t, 1);
00045
00046     /* Read tables... */
00047     sprintf(ctl.tblbase, "%s", argv[2]);
00048     ctl.tblfmt = atoi(argv[3]);
00049     read_tbl(&ctl, tbl);
00050
00051     /* Write tables... */
00052     sprintf(ctl.tblbase, "%s", argv[4]);
00053     ctl.tblfmt = atoi(argv[5]);
00054     write_tbl(&ctl, tbl);
00055
00056     /* Free... */
00057     free(tbl);
00058
00059     return EXIT_SUCCESS;
00060 }
  
```

## 5.37 tblgen.c File Reference

Prepape look-up tables from monochromatic absorption spectra.

```
#include "jurassic.h"
```

### Macros

- `#define MAXNF 10000`
- `#define MAXNPTS 10000000`
- `#define MAXLINE 100000`

### Functions

- `int main (int argc, char *argv[])`

#### 5.37.1 Detailed Description

Prepape look-up tables from monochromatic absorption spectra.

Definition in file [tblgen.c](#).

#### 5.37.2 Macro Definition Documentation

##### 5.37.2.1 MAXNF `#define MAXNF 10000`

Definition at line [32](#) of file [tblgen.c](#).

##### 5.37.2.2 MAXNPTS `#define MAXNPTS 10000000`

Definition at line [35](#) of file [tblgen.c](#).

##### 5.37.2.3 MAXLINE `#define MAXLINE 100000`

Definition at line [38](#) of file [tblgen.c](#).

#### 5.37.3 Function Documentation

```

5.37.3.1 main() int main (
                int argc,
                char * argv[] )

```

Definition at line 44 of file [tblgen.c](#).

```

00046     {
00047
00048     FILE *in;
00049
00050     static char line[MAXLINE], *tok;
00051
00052     static double dnu, abs[MAXNPTS], epsold, f, filt[MAXNF],
00053         nu, nu0, nul, nuf[MAXNF], press, r0, temp, u;
00054
00055     static int i, i0, idx, nf, npts;
00056
00057     /* Read command line arguments... */
00058     if (argc != 5)
00059         ERRMSG("Give parameters: <press> <temp> <spec> <filter>");
00060     sscanf(argv[1], "%lg", &press);
00061     sscanf(argv[2], "%lg", &temp);
00062
00063     /* Compute column density [molec/cm^2] (1 km path length, 1 ppmv)... */
00064     u = 1e-6 * press * 100 / (1.380658e-23 * temp) * 1000 / 1e4;
00065
00066     /* Read filter function... */
00067     if (!(in = fopen(argv[4], "r")))
00068         ERRMSG("Cannot open filter file!");
00069     while (fgets(line, MAXLINE, in))
00070         if (sscanf(line, "%lg %lg", &nuf[nf], &filt[nf]) == 2)
00071             if (++nf >= MAXNF)
00072                 ERRMSG("Too many points in filter function");
00073     fclose(in);
00074
00075     /* Read spectrum... */
00076     if (!(in = fopen(argv[3], "r")))
00077         ERRMSG("Cannot open spectrum!");
00078     if (!fgets(line, MAXLINE, in))
00079         ERRMSG("Error while reading spectrum!");
00080     if (!fgets(line, MAXLINE, in))
00081         ERRMSG("Error while reading spectrum!");
00082     if (!fgets(line, MAXLINE, in))
00083         ERRMSG("Error while reading spectrum!");
00084     if (!fgets(line, MAXLINE, in))
00085         ERRMSG("Error while reading spectrum!");
00086     sscanf(line, "%d %lg %lg %lg", &npts, &nu0, &dnu, &nul);
00087     if (npts > MAXNPTS)
00088         ERRMSG("Too many points in optical depth spectrum!");
00089     i = 0;
00090     while (fgets(line, MAXLINE, in)) {
00091         if ((tok = strtok(line, " \t\n")) != NULL) {
00092             sscanf(tok, "%lg", &abs[i]);
00093             abs[i] /= u;
00094             i++;
00095         }
00096         while ((tok = strtok(NULL, " \t\n")) != NULL) {
00097             sscanf(tok, "%lg", &abs[i]);
00098             abs[i] /= u;
00099             i++;
00100         }
00101     }
00102     fclose(in);
00103
00104     /* Set grid spacing... */
00105     dnu = (nul - nu0) / ((double) npts - 1.0);
00106     r0 = (nuf[0] - nu0) / (nul - nu0) * (double) npts;
00107     i0 = (int) r0;
00108
00109     /* Loop over column densities... */
00110     for (u = 1.0; u <= 1e30; u *= 1.122) {
00111
00112         /* Integrate... */
00113         double epssum = 0, fsum = 0;
00114         for (i = i0; i < npts; i++) {
00115             nu = nu0 + dnu * (double) i;
00116             if (nu < nuf[0])
00117                 continue;
00118             else if (nu > nuf[nf - 1])
00119                 break;
00120             else {
00121                 if (nu < nuf[idx] || nu > nuf[idx + 1])
00122                     idx = locate_irr(nuf, nf, nu);
00123                 f = LIN(nuf[idx], filt[idx], nuf[idx + 1], filt[idx + 1], nu);
00124                 fsum += f;
00125                 epssum += f * exp(-abs[i] * u);

```

```

00126     }
00127     }
00128     epssum = 1 - epssum / fsum;
00129
00130     /* Write output... */
00131     if (epssum >= 1e-6 && epssum <= 0.999999 && epssum > epsold)
00132         printf("%g %g %g %g\n", press, temp, u, epssum);
00133     epsold = epssum;
00134
00135     /* Check for termination... */
00136     if (epssum > 0.999999)
00137         return EXIT_SUCCESS;
00138 }
00139
00140 return EXIT_SUCCESS;
00141 }

```

Here is the call graph for this function:



## 5.38 tblgen.c

```

00001 /*
00002  This file is part of JURASSIC.
00003
00004  JURASSIC is free software: you can redistribute it and/or modify
00005  it under the terms of the GNU General Public License as published by
00006  the Free Software Foundation, either version 3 of the License, or
00007  (at your option) any later version.
00008
00009  JURASSIC is distributed in the hope that it will be useful,
00010  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  GNU General Public License for more details.
00013
00014  You should have received a copy of the GNU General Public License
00015  along with JURASSIC. If not, see <http://www.gnu.org/licenses/>.
00016
00017  Copyright (C) 2003-2021 Forschungszentrum Juelich GmbH
00018 */
00019
00025 #include "jurassic.h"
00026
00027 /* -----
00028  Dimensions...
00029  ----- */
00030
00031 /* Maximum number of grid points for filter files: */
00032 #define MAXNF 10000
00033
00034 /* Maximum number of grid points for spectra: */
00035 #define MAXNPTS 10000000
00036
00037 /* Maximum line length: */
00038 #define MAXLINE 100000
00039
00040 /* -----
00041  Main...
00042  ----- */
00043
00044 int main(
00045     int argc,
00046     char *argv[]) {
00047     FILE *in;
00048
00049     static char line[MAXLINE], *tok;
00050
00051     static double dnu, abs[MAXNPTS], epsold, f, filt[MAXNF],

```

```

00053     nu, nu0, nul, nuf[MAXNF], press, r0, temp, u;
00054
00055     static int i, i0, idx, nf, npts;
00056
00057     /* Read command line arguments... */
00058     if (argc != 5)
00059         ERRMSG("Give parameters: <press> <temp> <spec> <filter>");
00060     sscanf(argv[1], "%lg", &press);
00061     sscanf(argv[2], "%lg", &temp);
00062
00063     /* Compute column density [molec/cm^2] (1 km path length, 1 ppmv)... */
00064     u = 1e-6 * press * 100 / (1.380658e-23 * temp) * 1000 / 1e4;
00065
00066     /* Read filter function... */
00067     if (!(in = fopen(argv[4], "r")))
00068         ERRMSG("Cannot open filter file!");
00069     while (fgets(line, MAXLINE, in))
00070         if (sscanf(line, "%lg %lg", &nuf[nf], &filt[nf]) == 2)
00071             if (++nf >= MAXNF)
00072                 ERRMSG("Too many points in filter function");
00073     fclose(in);
00074
00075     /* Read spectrum... */
00076     if (!(in = fopen(argv[3], "r")))
00077         ERRMSG("Cannot open spectrum!");
00078     if (!fgets(line, MAXLINE, in))
00079         ERRMSG("Error while reading spectrum!");
00080     if (!fgets(line, MAXLINE, in))
00081         ERRMSG("Error while reading spectrum!");
00082     if (!fgets(line, MAXLINE, in))
00083         ERRMSG("Error while reading spectrum!");
00084     if (!fgets(line, MAXLINE, in))
00085         ERRMSG("Error while reading spectrum!");
00086     sscanf(line, "%d %lg %lg %lg", &npts, &nu0, &dnu, &nul);
00087     if (npts > MAXNPTS)
00088         ERRMSG("Too many points in optical depth spectrum!");
00089     i = 0;
00090     while (fgets(line, MAXLINE, in)) {
00091         if ((tok = strtok(line, " \t\n")) != NULL) {
00092             sscanf(tok, "%lg", &abs[i]);
00093             abs[i] /= u;
00094             i++;
00095         }
00096         while ((tok = strtok(NULL, " \t\n")) != NULL) {
00097             sscanf(tok, "%lg", &abs[i]);
00098             abs[i] /= u;
00099             i++;
00100         }
00101     }
00102     fclose(in);
00103
00104     /* Set grid spacing... */
00105     dnu = (nul - nu0) / ((double) npts - 1.0);
00106     r0 = (nuf[0] - nu0) / (nul - nu0) * (double) npts;
00107     i0 = (int) r0;
00108
00109     /* Loop over column densities... */
00110     for (u = 1.0; u <= 1e30; u *= 1.122) {
00111
00112         /* Integrate... */
00113         double epssum = 0, fsum = 0;
00114         for (i = i0; i < npts; i++) {
00115             nu = nu0 + dnu * (double) i;
00116             if (nu < nuf[0])
00117                 continue;
00118             else if (nu > nuf[nf - 1])
00119                 break;
00120             else {
00121                 if (nu < nuf[idx] || nu > nuf[idx + 1])
00122                     idx = locate_irr(nuf, nf, nu);
00123                 f = LIN(nuf[idx], filt[idx], nuf[idx + 1], filt[idx + 1], nu);
00124                 fsum += f;
00125                 epssum += f * exp(-abs[i] * u);
00126             }
00127         }
00128         epssum = 1 - epssum / fsum;
00129
00130         /* Write output... */
00131         if (epssum >= 1e-6 && epssum <= 0.999999 && epssum > epsold)
00132             printf("%g %g %g %g\n", press, temp, u, epssum);
00133         epsold = epssum;
00134
00135         /* Check for termination... */
00136         if (epssum > 0.999999)
00137             return EXIT_SUCCESS;
00138     }
00139

```

```
00140     return EXIT_SUCCESS;
00141 }
```

## 5.39 time2jsec.c File Reference

Convert date to Julian seconds.

```
#include "jurassic.h"
```

### Functions

- int [main](#) (int argc, char \*argv[])

#### 5.39.1 Detailed Description

Convert date to Julian seconds.

Definition in file [time2jsec.c](#).

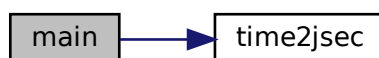
#### 5.39.2 Function Documentation

**5.39.2.1 main()** int main (  
    int argc,  
    char \* argv[] )

Definition at line 27 of file [time2jsec.c](#).

```
00029     {
00030
00031     double jsec;
00032
00033     /* Check arguments... */
00034     if (argc < 8)
00035         ERRMSG("Give parameters: <year> <mon> <day> <hour> <min> <sec> <remain>");
00036
00037     /* Read arguments... */
00038     int year = atoi(argv[1]);
00039     int mon = atoi(argv[2]);
00040     int day = atoi(argv[3]);
00041     int hour = atoi(argv[4]);
00042     int min = atoi(argv[5]);
00043     int sec = atoi(argv[6]);
00044     double remain = atof(argv[7]);
00045
00046     /* Convert... */
00047     time2jsec(year, mon, day, hour, min, sec, remain, &jsec);
00048     printf("%.2f\n", jsec);
00049
00050     return EXIT_SUCCESS;
00051 }
```

Here is the call graph for this function:



## 5.40 time2jsec.c

```
00001 /*
00002  This file is part of JURASSIC.
00003
00004  JURASSIC is free software: you can redistribute it and/or modify
00005  it under the terms of the GNU General Public License as published by
00006  the Free Software Foundation, either version 3 of the License, or
00007  (at your option) any later version.
00008
00009  JURASSIC is distributed in the hope that it will be useful,
00010  but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  GNU General Public License for more details.
00013
00014  You should have received a copy of the GNU General Public License
00015  along with JURASSIC. If not, see <http://www.gnu.org/licenses/>.
00016
00017  Copyright (C) 2003-2021 Forschungszentrum Juelich GmbH
00018 */
00019
00025 #include "jurassic.h"
00026
00027 int main(
00028     int argc,
00029     char *argv[]) {
00030
00031     double jsec;
00032
00033     /* Check arguments... */
00034     if (argc < 8)
00035         ERRMSG("Give parameters: <year> <mon> <day> <hour> <min> <sec> <remain>");
00036
00037     /* Read arguments... */
00038     int year = atoi(argv[1]);
00039     int mon = atoi(argv[2]);
00040     int day = atoi(argv[3]);
00041     int hour = atoi(argv[4]);
00042     int min = atoi(argv[5]);
00043     int sec = atoi(argv[6]);
00044     double remain = atof(argv[7]);
00045
00046     /* Convert... */
00047     time2jsec(year, mon, day, hour, min, sec, remain, &jsec);
00048     printf("%.2f\n", jsec);
00049
00050     return EXIT_SUCCESS;
00051 }
```





## Index

ails  
    filter.c, 36  
ALLOC  
    jurassic.h, 221  
analyze\_avk  
    retrieval.c, 341  
analyze\_avk\_quantity  
    retrieval.c, 342  
atm2x  
    jurassic.c, 68  
    jurassic.h, 232  
atm2x\_help  
    jurassic.c, 69  
    jurassic.h, 233  
atm\_t, 3  
    cldz, 6  
    clk, 6  
    clz, 5  
    k, 5  
    lat, 5  
    lon, 4  
    np, 4  
    p, 5  
    q, 5  
    sfeqs, 6  
    sfp, 6  
    sft, 6  
    sfz, 6  
    t, 5  
    time, 4  
    z, 4  
  
brightness  
    jurassic.c, 69  
    jurassic.h, 234  
brightness.c, 31, 32  
    main, 31  
  
C1  
    jurassic.h, 225  
C2  
    jurassic.h, 226  
call\_formod  
    formod.c, 41  
cart2geo  
    jurassic.c, 69  
    jurassic.h, 234  
cldz  
    atm\_t, 6  
climatology  
    jurassic.c, 70  
    jurassic.h, 234  
climatology.c, 33, 35  
    main, 34  
clk  
    atm\_t, 6  
  
cldz  
    atm\_t, 10  
clz  
    atm\_t, 5  
conv\_dmin  
    ret\_t, 25  
conv\_itmax  
    ret\_t, 24  
copy\_atm  
    jurassic.c, 103  
    jurassic.h, 268  
copy\_obs  
    jurassic.c, 104  
    jurassic.h, 269  
cost\_function  
    retrieval.c, 342  
ctl\_t, 7  
    cldz, 10  
    ctm\_co2, 11  
    ctm\_h2o, 11  
    ctm\_n2, 12  
    ctm\_o2, 12  
    emitter, 9  
    formod, 15  
    fov, 12  
    hyd, 11  
    ncl, 10  
    nd, 9  
    ng, 9  
    nsf, 10  
    nu, 9  
    nw, 9  
    rayds, 12  
    raydz, 12  
    refrac, 12  
    ret\_cldz, 14  
    ret\_clk, 14  
    ret\_clz, 14  
    ret\_sfeqs, 15  
    ret\_sfp, 15  
    ret\_sft, 15  
    ret\_sfz, 14  
    retk\_zmax, 14  
    retk\_zmin, 14  
    retp\_zmax, 13  
    retp\_zmin, 13  
    retq\_zmax, 13  
    retq\_zmin, 13  
    rett\_zmax, 13  
    rett\_zmin, 13  
    rfmbin, 16  
    rfmhit, 16  
    rfmxsc, 16  
    sfnu, 10  
    sfsza, 11

- sftype, 10
- tblbase, 11
- tblfmt, 11
- window, 10
- write\_bbt, 15
- write\_matrix, 15
- ctm\_co2
  - ctl\_t, 11
- ctm\_h2o
  - ctl\_t, 11
- ctm\_n2
  - ctl\_t, 12
- ctm\_o2
  - ctl\_t, 12
- ctmco2
  - jurassic.c, 79
  - jurassic.h, 244
- ctmh2o
  - jurassic.c, 89
  - jurassic.h, 254
- ctmn2
  - jurassic.c, 101
  - jurassic.h, 266
- ctmo2
  - jurassic.c, 102
  - jurassic.h, 267
- dir
  - ret\_t, 24
- DIST
  - jurassic.h, 221
- DIST2
  - jurassic.h, 221
- DOTP
  - jurassic.h, 221
- ds
  - los\_t, 19
- emitter
  - ctl\_t, 9
- eps
  - los\_t, 19
  - tbl\_t, 30
- err\_ana
  - ret\_t, 25
- err\_cldz
  - ret\_t, 27
- err\_clk
  - ret\_t, 28
- err\_clz
  - ret\_t, 27
- err\_formod
  - ret\_t, 25
- err\_k
  - ret\_t, 27
- err\_k\_ch
  - ret\_t, 27
- err\_k\_cz
  - ret\_t, 27
- err\_noise
  - ret\_t, 25
- err\_press
  - ret\_t, 25
- err\_press\_ch
  - ret\_t, 26
- err\_press\_cz
  - ret\_t, 25
- err\_q
  - ret\_t, 26
- err\_q\_ch
  - ret\_t, 27
- err\_q\_cz
  - ret\_t, 26
- err\_sfeps
  - ret\_t, 28
- err\_sfp
  - ret\_t, 28
- err\_sft
  - ret\_t, 28
- err\_sfz
  - ret\_t, 28
- err\_temp
  - ret\_t, 26
- err\_temp\_ch
  - ret\_t, 26
- err\_temp\_cz
  - ret\_t, 26
- ERRMSG
  - jurassic.h, 224
- EXP
  - jurassic.h, 221
- filter.c, 36, 39
  - ails, 36
  - main, 37
- find\_emitter
  - jurassic.c, 105
  - jurassic.h, 269
- formod
  - ctl\_t, 15
  - jurassic.c, 105
  - jurassic.h, 270
- formod.c, 40, 46
  - call\_formod, 41
  - main, 44
- formod\_continua
  - jurassic.c, 106
  - jurassic.h, 271
- formod\_fov
  - jurassic.c, 107
  - jurassic.h, 272
- formod\_pencil
  - jurassic.c, 109
  - jurassic.h, 274
- formod\_rfm
  - jurassic.c, 111
  - jurassic.h, 276
- formod\_srcfunc

- jurassic.c, [114](#)
  - jurassic.h, [279](#)
- fov
  - ctl\_t, [12](#)
- FREAD
  - jurassic.h, [222](#)
- FWRITE
  - jurassic.h, [222](#)
- G0
  - jurassic.h, [226](#)
- geo2cart
  - jurassic.c, [114](#)
  - jurassic.h, [279](#)
- H0
  - jurassic.h, [226](#)
- hydrostatic
  - jurassic.c, [114](#)
  - jurassic.h, [279](#)
- hydrostatic.c, [50](#), [51](#)
  - main, [50](#)
- hydz
  - ctl\_t, [11](#)
- idx2name
  - jurassic.c, [116](#)
  - jurassic.h, [281](#)
- IDXCLDZ
  - jurassic.h, [231](#)
- IDXCLK
  - jurassic.h, [232](#)
- IDXCLZ
  - jurassic.h, [231](#)
- IDXK
  - jurassic.h, [231](#)
- IDXP
  - jurassic.h, [231](#)
- IDXQ
  - jurassic.h, [231](#)
- IDXSFEPS
  - jurassic.h, [232](#)
- IDXSFP
  - jurassic.h, [232](#)
- IDXSFT
  - jurassic.h, [232](#)
- IDXSFZ
  - jurassic.h, [232](#)
- IDXT
  - jurassic.h, [231](#)
- init\_srcfunc
  - jurassic.c, [116](#)
  - jurassic.h, [281](#)
- interpolate.c, [52](#), [53](#)
  - main, [52](#)
- intpol\_atm
  - jurassic.c, [117](#)
  - jurassic.h, [282](#)
- intpol\_tbl
  - jurassic.c, [118](#)
  - jurassic.h, [283](#)
- intpol\_tbl\_eps
  - jurassic.c, [119](#)
  - jurassic.h, [284](#)
- intpol\_tbl\_u
  - jurassic.c, [120](#)
  - jurassic.h, [285](#)
- invert.c, [54](#), [59](#)
  - main, [55](#)
  - NLMAX, [55](#)
  - NMAX, [55](#)
- jsec2time
  - jurassic.c, [121](#)
  - jurassic.h, [286](#)
- jsec2time.c, [64](#), [65](#)
  - main, [64](#)
- jurassic.c, [65](#), [150](#)
  - atm2x, [68](#)
  - atm2x\_help, [69](#)
  - brightness, [69](#)
  - cart2geo, [69](#)
  - climatology, [70](#)
  - copy\_atm, [103](#)
  - copy\_obs, [104](#)
  - ctmco2, [79](#)
  - ctmh2o, [89](#)
  - ctmn2, [101](#)
  - ctmo2, [102](#)
  - find\_emitter, [105](#)
  - formod, [105](#)
  - formod\_continua, [106](#)
  - formod\_fov, [107](#)
  - formod\_pencil, [109](#)
  - formod\_rfm, [111](#)
  - formod\_srcfunc, [114](#)
  - geo2cart, [114](#)
  - hydrostatic, [114](#)
  - idx2name, [116](#)
  - init\_srcfunc, [116](#)
  - intpol\_atm, [117](#)
  - intpol\_tbl, [118](#)
  - intpol\_tbl\_eps, [119](#)
  - intpol\_tbl\_u, [120](#)
  - jsec2time, [121](#)
  - kernel, [122](#)
  - locate\_irr, [123](#)
  - locate\_reg, [124](#)
  - locate\_tbl, [124](#)
  - obs2y, [125](#)
  - planck, [125](#)
  - raytrace, [125](#)
  - read\_atm, [129](#)
  - read\_ctl, [130](#)
  - read\_matrix, [132](#)
  - read\_obs, [132](#)
  - read\_obs\_rfm, [133](#)
  - read\_rfm\_spec, [134](#)

- read\_shape, 135
- read\_tbl, 135
- refractivity, 137
- scan\_ctl, 137
- sza, 138
- tangent\_point, 139
- time2jsec, 140
- timer, 140
- write\_atm, 141
- write\_atm\_rfm, 142
- write\_matrix, 143
- write\_obs, 145
- write\_shape, 146
- write\_tbl, 147
- x2atm, 148
- x2atm\_help, 149
- y2obs, 149
- jurassic.h, 215, 315
  - ALLOC, 221
  - atm2x, 232
  - atm2x\_help, 233
  - brightness, 234
  - C1, 225
  - C2, 226
  - cart2geo, 234
  - climatology, 234
  - copy\_atm, 268
  - copy\_obs, 269
  - ctmco2, 244
  - ctmh2o, 254
  - ctmn2, 266
  - ctmo2, 267
  - DIST, 221
  - DIST2, 221
  - DOTP, 221
  - ERRMSG, 224
  - EXP, 221
  - find\_emitter, 269
  - formod, 270
  - formod\_continua, 271
  - formod\_fov, 272
  - formod\_pencil, 274
  - formod\_rfm, 276
  - formod\_srcfunc, 279
  - FREAD, 222
  - FWRITE, 222
  - G0, 226
  - geo2cart, 279
  - H0, 226
  - hydrostatic, 279
  - idx2name, 281
  - IDXCLDZ, 231
  - IDXCLK, 232
  - IDXCLZ, 231
  - IDXK, 231
  - IDXP, 231
  - IDXQ, 231
  - IDXSFEPS, 232
  - IDXSFP, 232
  - IDXSFT, 232
  - IDXSFZ, 232
  - IDXT, 231
  - init\_srcfunc, 281
  - intpol\_atm, 282
  - intpol\_tbl, 283
  - intpol\_tbl\_eps, 284
  - intpol\_tbl\_u, 285
  - jsec2time, 286
  - KB, 226
  - kernel, 287
  - LEN, 228
  - LIN, 222
  - locate\_irr, 288
  - locate\_reg, 289
  - locate\_tbl, 289
  - LOG, 224
  - LOGLEV, 224
  - M, 229
  - ME, 227
  - N, 229
  - NA, 226
  - NCL, 227
  - ND, 227
  - NFOV, 229
  - NG, 228
  - NLOS, 229
  - NORM, 223
  - NP, 228
  - NQ, 229
  - NR, 228
  - NSF, 228
  - NSHAPE, 229
  - NW, 228
  - obs2y, 290
  - P0, 226
  - planck, 290
  - POW2, 223
  - POW3, 223
  - PRINT, 225
  - raytrace, 290
  - RE, 227
  - read\_atm, 294
  - read\_ctl, 295
  - read\_matrix, 297
  - read\_obs, 297
  - read\_obs\_rfm, 298
  - read\_rfm\_spec, 299
  - read\_shape, 300
  - read\_tbl, 300
  - refractivity, 302
  - RFMLINE, 230
  - RFMNPTS, 230
  - RI, 227
  - scan\_ctl, 302
  - sza, 303
  - T0, 227

- tangent\_point, 304
- TBLNP, 230
- TBLNS, 230
- TBLNT, 230
- TBLNU, 230
- time2jsec, 305
- TIMER, 223
- timer, 305
- TMAX, 225
- TMIN, 225
- TOK, 223
- TSUN, 225
- WARN, 224
- write\_atm, 306
- write\_atm\_rfm, 307
- write\_matrix, 308
- write\_obs, 310
- write\_shape, 311
- write\_tbl, 312
- x2atm, 313
- x2atm\_help, 314
- y2obs, 314
- k
  - atm\_t, 5
  - los\_t, 18
- KB
  - jurassic.h, 226
- kernel
  - jurassic.c, 122
  - jurassic.h, 287
- kernel.c, 324, 326
  - main, 324
- kernel\_recomp
  - ret\_t, 24
- lat
  - atm\_t, 5
  - los\_t, 18
- LEN
  - jurassic.h, 228
- limb.c, 327, 328
  - main, 327
- LIN
  - jurassic.h, 222
- locate\_irr
  - jurassic.c, 123
  - jurassic.h, 288
- locate\_reg
  - jurassic.c, 124
  - jurassic.h, 289
- locate\_tbl
  - jurassic.c, 124
  - jurassic.h, 289
- LOG
  - jurassic.h, 224
- LOGLEV
  - jurassic.h, 224
- lon
  - atm\_t, 4
  - los\_t, 18
- los\_t, 16
  - ds, 19
  - eps, 19
  - k, 18
  - lat, 18
  - lon, 18
  - np, 17
  - p, 18
  - q, 18
  - sfepts, 19
  - sft, 19
  - src, 19
  - t, 18
  - u, 19
  - z, 17
- M
  - jurassic.h, 229
- main
  - brightness.c, 31
  - climatology.c, 34
  - filter.c, 37
  - formod.c, 44
  - hydrostatic.c, 50
  - interpolate.c, 52
  - invert.c, 55
  - jsec2time.c, 64
  - kernel.c, 324
  - limb.c, 327
  - nadir.c, 329
  - obs2spec.c, 331
  - planck.c, 334
  - raytrace.c, 337
  - retrieval.c, 353
  - tblfmt.c, 366
  - tblgen.c, 368
  - time2jsec.c, 372
- matrix\_invert
  - retrieval.c, 343
- matrix\_product
  - retrieval.c, 343
- MAXLINE
  - tblgen.c, 368
- MAXNF
  - tblgen.c, 368
- MAXNPTS
  - tblgen.c, 368
- ME
  - jurassic.h, 227
- N
  - jurassic.h, 229
- NA
  - jurassic.h, 226
- nadir.c, 329, 330
  - main, 329
- NCL

- jurassic.h, 227
- ncl
  - ctl\_t, 10
- ND
  - jurassic.h, 227
- nd
  - ctl\_t, 9
- NFOV
  - jurassic.h, 229
- NG
  - jurassic.h, 228
- ng
  - ctl\_t, 9
- NLMAX
  - invert.c, 55
- NLOS
  - jurassic.h, 229
- NMAX
  - invert.c, 55
- NORM
  - jurassic.h, 223
- NP
  - jurassic.h, 228
- np
  - atm\_t, 4
  - los\_t, 17
  - tbl\_t, 29
- NQ
  - jurassic.h, 229
- NR
  - jurassic.h, 228
- nr
  - obs\_t, 21
- NSF
  - jurassic.h, 228
- nsf
  - ctl\_t, 10
- NSHAPE
  - jurassic.h, 229
- nt
  - tbl\_t, 29
- nu
  - ctl\_t, 9
  - tbl\_t, 30
- NW
  - jurassic.h, 228
- nw
  - ctl\_t, 9
- obs2spec.c, 331, 333
  - main, 331
- obs2y
  - jurassic.c, 125
  - jurassic.h, 290
- obs\_t, 20
  - nr, 21
  - obslat, 21
  - obslon, 21
  - obsz, 21
- rad, 22
- tau, 22
- time, 21
- tplat, 22
- tplon, 22
- tpz, 22
- vplat, 22
- vplon, 21
- vpz, 21
- obslat
  - obs\_t, 21
- obslon
  - obs\_t, 21
- obsz
  - obs\_t, 21
- optimal\_estimation
  - retrieval.c, 344
- p
  - atm\_t, 5
  - los\_t, 18
  - tbl\_t, 30
- P0
  - jurassic.h, 226
- planck
  - jurassic.c, 125
  - jurassic.h, 290
- planck.c, 334, 335
  - main, 334
- POW2
  - jurassic.h, 223
- POW3
  - jurassic.h, 223
- PRINT
  - jurassic.h, 225
- q
  - atm\_t, 5
  - los\_t, 18
- rad
  - obs\_t, 22
- rayds
  - ctl\_t, 12
- raydz
  - ctl\_t, 12
- raytrace
  - jurassic.c, 125
  - jurassic.h, 290
- raytrace.c, 336, 338
  - main, 337
- RE
  - jurassic.h, 227
- read\_atm
  - jurassic.c, 129
  - jurassic.h, 294
- read\_ctl
  - jurassic.c, 130
  - jurassic.h, 295

- read\_matrix
  - jurassic.c, [132](#)
  - jurassic.h, [297](#)
- read\_obs
  - jurassic.c, [132](#)
  - jurassic.h, [297](#)
- read\_obs\_rfm
  - jurassic.c, [133](#)
  - jurassic.h, [298](#)
- read\_ret
  - retrieval.c, [348](#)
- read\_rfm\_spec
  - jurassic.c, [134](#)
  - jurassic.h, [299](#)
- read\_shape
  - jurassic.c, [135](#)
  - jurassic.h, [300](#)
- read\_tbl
  - jurassic.c, [135](#)
  - jurassic.h, [300](#)
- refrac
  - ctl\_t, [12](#)
- refractivity
  - jurassic.c, [137](#)
  - jurassic.h, [302](#)
- ret\_cldz
  - ctl\_t, [14](#)
- ret\_clk
  - ctl\_t, [14](#)
- ret\_clz
  - ctl\_t, [14](#)
- ret\_sfeps
  - ctl\_t, [15](#)
- ret\_sfp
  - ctl\_t, [15](#)
- ret\_sft
  - ctl\_t, [15](#)
- ret\_sfz
  - ctl\_t, [14](#)
- ret\_t, [23](#)
  - conv\_dmin, [25](#)
  - conv\_itmax, [24](#)
  - dir, [24](#)
  - err\_ana, [25](#)
  - err\_cldz, [27](#)
  - err\_clk, [28](#)
  - err\_clz, [27](#)
  - err\_formod, [25](#)
  - err\_k, [27](#)
  - err\_k\_ch, [27](#)
  - err\_k\_cz, [27](#)
  - err\_noise, [25](#)
  - err\_press, [25](#)
  - err\_press\_ch, [26](#)
  - err\_press\_cz, [25](#)
  - err\_q, [26](#)
  - err\_q\_ch, [27](#)
  - err\_q\_cz, [26](#)
  - err\_sfeps, [28](#)
  - err\_sfp, [28](#)
  - err\_sft, [28](#)
  - err\_sfz, [28](#)
  - err\_temp, [26](#)
  - err\_temp\_ch, [26](#)
  - err\_temp\_cz, [26](#)
  - kernel\_recomp, [24](#)
- retk\_zmax
  - ctl\_t, [14](#)
- retk\_zmin
  - ctl\_t, [14](#)
- retp\_zmax
  - ctl\_t, [13](#)
- retp\_zmin
  - ctl\_t, [13](#)
- retq\_zmax
  - ctl\_t, [13](#)
- retq\_zmin
  - ctl\_t, [13](#)
- retrieval.c, [340](#), [354](#)
  - analyze\_avk, [341](#)
  - analyze\_avk\_quantity, [342](#)
  - cost\_function, [342](#)
  - main, [353](#)
  - matrix\_invert, [343](#)
  - matrix\_product, [343](#)
  - optimal\_estimation, [344](#)
  - read\_ret, [348](#)
  - set\_cov\_apr, [349](#)
  - set\_cov\_meas, [351](#)
  - write\_stddev, [352](#)
- rett\_zmax
  - ctl\_t, [13](#)
- rett\_zmin
  - ctl\_t, [13](#)
- rfmbin
  - ctl\_t, [16](#)
- rfmhit
  - ctl\_t, [16](#)
- RFMLINE
  - jurassic.h, [230](#)
- RFMNPTS
  - jurassic.h, [230](#)
- rfmxsc
  - ctl\_t, [16](#)
- RI
  - jurassic.h, [227](#)
- scan\_ctl
  - jurassic.c, [137](#)
  - jurassic.h, [302](#)
- set\_cov\_apr
  - retrieval.c, [349](#)
- set\_cov\_meas
  - retrieval.c, [351](#)
- sfeps
  - atm\_t, [6](#)
  - los\_t, [19](#)



- sfnu
  - ctl\_t, 10
- sfp
  - atm\_t, 6
- sfsza
  - ctl\_t, 11
- sft
  - atm\_t, 6
  - los\_t, 19
- sftype
  - ctl\_t, 10
- sfz
  - atm\_t, 6
- sr
  - tbl\_t, 31
- src
  - los\_t, 19
- st
  - tbl\_t, 30
- sza
  - jurassic.c, 138
  - jurassic.h, 303
- t
  - atm\_t, 5
  - los\_t, 18
  - tbl\_t, 30
- T0
  - jurassic.h, 227
- tangent\_point
  - jurassic.c, 139
  - jurassic.h, 304
- tau
  - obs\_t, 22
- tbl\_t, 29
  - eps, 30
  - np, 29
  - nt, 29
  - nu, 30
  - p, 30
  - sr, 31
  - st, 30
  - t, 30
  - u, 30
- tblbase
  - ctl\_t, 11
- tblfmt
  - ctl\_t, 11
- tblfmt.c, 366, 367
  - main, 366
- tblgen.c, 368, 370
  - main, 368
  - MAXLINE, 368
  - MAXNF, 368
  - MAXNPTS, 368
- TBLNP
  - jurassic.h, 230
- TBLNS
  - jurassic.h, 230
- TBLNT
  - jurassic.h, 230
- TBLNU
  - jurassic.h, 230
- time
  - atm\_t, 4
  - obs\_t, 21
- time2jsec
  - jurassic.c, 140
  - jurassic.h, 305
- time2jsec.c, 372, 373
  - main, 372
- TIMER
  - jurassic.h, 223
- timer
  - jurassic.c, 140
  - jurassic.h, 305
- TMAX
  - jurassic.h, 225
- TMIN
  - jurassic.h, 225
- TOK
  - jurassic.h, 223
- tplat
  - obs\_t, 22
- tplon
  - obs\_t, 22
- tpz
  - obs\_t, 22
- TSUN
  - jurassic.h, 225
- u
  - los\_t, 19
  - tbl\_t, 30
- vplat
  - obs\_t, 22
- vplon
  - obs\_t, 21
- vpz
  - obs\_t, 21
- WARN
  - jurassic.h, 224
- window
  - ctl\_t, 10
- write\_atm
  - jurassic.c, 141
  - jurassic.h, 306
- write\_atm\_rfm
  - jurassic.c, 142
  - jurassic.h, 307
- write\_bbt
  - ctl\_t, 15
- write\_matrix
  - ctl\_t, 15
  - jurassic.c, 143
  - jurassic.h, 308

write\_obs  
    jurassic.c, [145](#)  
    jurassic.h, [310](#)  
write\_shape  
    jurassic.c, [146](#)  
    jurassic.h, [311](#)  
write\_stddev  
    retrieval.c, [352](#)  
write\_tbl  
    jurassic.c, [147](#)  
    jurassic.h, [312](#)  
  
x2atm  
    jurassic.c, [148](#)  
    jurassic.h, [313](#)  
x2atm\_help  
    jurassic.c, [149](#)  
    jurassic.h, [314](#)  
  
y2obs  
    jurassic.c, [149](#)  
    jurassic.h, [314](#)  
  
z  
    atm\_t, [4](#)  
    los\_t, [17](#)